TM 1003/002/00

Milestone II

160 A Diagnostic Program (SFCHEX)

TM-1003/002/00

TECHNICAL MEMORANDUM

(TM Series)

ASTIA AVAILABILITY NOTICE

Qualified requesters may obtain copies of this report from ASTIA.

This document was produced by SDC in performance of contract AF 19(628)-1648, Space Systems Division Program, for Space Systems Division, AFSC.

Milestone 11

160-A Diagnostic Program (SFCHEX)

By

Utility Section

12 February 1963

Approved

J. B. Munson

SYSTEM DEVELOPMENT CORPORATION
2500 COLORADO AVE.
SANTA MONICA
CALIFORNIA

PREFACE

Acknowledgement is made of the Milestone 11 document, produced by Mellonics S-D, Incorporated, from which the contents of this document were drawn.

TABLE OF CONTENTS

ILLUSTRATIONS

IDENTIFICATION

Name:
160-A Diagnostic Program (SFCHEX) - Ident 02C, Mod AA

Program Authors:
G. Sanford and S. Lynch, Mellonics System Development, Inc.
18 January 1963

Documented:
Mellonics Systems Development Inc.
18 January 1963

Modified:
M. J. Sweeney, System Development Corporation
1 February 1963

Edited:
R. C. Wise, System Development Corporation
12 February 1963

1.0 The SFCHEX Program

For many years it has been the practice of computer systems to supply a certain amount of diagnostic control in the assembly or compilation of computer programs. This control generally consists in the identification of duplicate tags, illegal instructions, invalid logical procedures and the like. However, this type of diagnostic aid is designed to assist the programmer only up to that time at which the computer is able to interpret and execute the instruction sequences. When the programmer begins to check out the computer program for the desired results, he is often left with little aid in the form of diagnostic routines. Even those which are available (such as dumps and trace routines) are not available at any selected point in the user's program unless the user modify his program by inserting some sort of calling sequence.

It is this situation that SFCHEX is intended to remedy in the case of the Bird Buffer System. SFCHEX will allow the user to dump any portion of his program at any time. He will be able to modify parameters or instructions as he desires. He may jump from point to point in his program. Annotations for output (e.g. dumps) can be generated. All these can be effected without the user modifying any portion of his program. The ultimate speed up in the checkout of programs and the added efficiency, in the fact that numerous assemblies and compilations to add or remove diagnostic dumps are no longer needed, is evident.

SFCHEX performs this function by accepting directions (i.e., commands) via the typewriter or on-line card reader. These commands instruct SFCHEX as to where and when corrections, dumps or annotations are to be made. The key command for SFCHEX is the TRAP Command. It enables the user to iterate a certain set of commands as many times as he wishes and after the final iteration, the "flushing" (i.e., elimination) of the TRAP Command will result in an interrogation of the typewriter (or card reader) for more commands.

Before explaining the method and philosophy of the program further, we shall now consider the input formats (Section 2.0), the output formats (Section 3.0) and the operating instructions (Section 4.0). Section 5.0 (SFCHEC Capability) will then develop further the method of SFCHEX and discuss the use and strategy of the various commands with emphasis on the TRAP Command.

Section 6.0 will be the listing of the validation and checkout procedures and Section 7.0 is a description of each subroutine and logical section of SFCHEX including both descriptions and flowcharts.

2.0 INPUT

2.1 Input Modes

SFCHEX allows two modes of input. Commands may be entered via the typewriter or the on-line card reader. SFCHEX will request the input option by printing on the typewriter: MODE IN.

The user will then indicate his option by typing a "T" for typewriter input or a "C" for punched card input. Input formats are the same for both modes of input. In both types of input a period will be used to terminate the command. In the typewriter input mode SFCHEX causes a carriage return and types an "*" whenever it is ready to accept another command. If the final period for a command has not yet been typed the user can delete the command by causing a carriage return. In the on-line card read mode SFCHEX reads one card each time it is prepared to accept another command.

2.2 Input Commands

There are twelve different commands which SFCHEX will accept and execute. Each command is identified by a letter which is the initial letter of the command name. The following is a list of the command names and a description of the function which each will perform. Please note all addresses and parameters are entered in octal.

2.2.1 Command Types

2.2.1.1 Begin Command

This command is used to indicate to the SFCHEX program that all of the initial commands have been entered. SFCHEX will transfer to the address contained in the Begin Command.(This address will normally be the initial addresss of the user's program).

The Begin Command may also be used to transfer to unconnected points in the user's program. This will be useful when the user's program is not able to run to completion, since several sections of the program may be reached by the command.

2.2.1.2 Correct Command

This command allows the user to enter octal correction and parameter modifications into any location in the computer. A maximum of eleven corrections is allowed in each Correct Command.

2.2.1.3 Dump Command

This command will cause an octal dump of the computer memory between the limits prescribed by the Dump Command. The dump will be generated on the on-line printer or magnetic tape according to the user's expressed output option (see Section 3.1).

2.2.1.4 End Command

This is one of two commands which is used to terminate command input to SFCHEX. SFCHEX will continue to execute all the commands which were entered and the user's program will be allowed to run until it reaches its final stop (or some error stop).

2.2.1.5 Header Command

This command is used to generate BCD annotation. Up to seventy-six characters may be input per command. These characters will then appear on the printer or magnetic tape according to the output option selected at the beginning of the run.

2.2.1.6 Locate Command

This command is necessary if the user intends to enter any Trap (2.2.1.10) or Snap (2.2.1.9) commands. The Locate command contains the initial address of a nine cell sequence in the user's relative bank, in which SFCHEX may store a subroutine. This subroutine (RECOV) will be the communications link between the user's program and SFCHEX, that is, each time SFCHEX sets a jumpout in the user's program, the jump will be to RECOV. RECOV will save the A-Register and the return address before transferring to SFCHEX in another bank.

NOTE: The Locate Command must be entered prior to any Snap, Trap or Unsnap commands.

2.2.1.7 Page Command

This command will cause a page eject in the output listing.

2.2.1.8 Run Command

This command is used to indicate to SFCHEX that no more commands will be entered at this time. This will cause SFCHEX to process the current commands until one of the Trap commands terminates. At this time SFCHEX will return for more input commands.

2.2.1.9 Snap Command

The Snap command is actually a specialized Dump command, which is executed each time the computer encounters a prescribed cell in the user's program. This specialized dump will consist of the bank control settings, the contents of the A-Register, the contents of the Buffer Entrance Register, and the first 77_8 cells of a prescribed bank (or all four banks). The Snap command will remain in effect until a corresponding Unsnap command (2.2.1.11) is entered and executed.

2.2.1.10 Trap Command

The Trap command performs three functions.

a. Iterate Dump, Heading, Correct, Mnemonic, and Page commands.
b. Delay the execution of Correct, Dump, Heading, Page, Mnemonic, Snap, and Unsnap commands until some strategic location is reached in the user's program.
c. Return for more command input at appropriate times.

The Trap command is set up in the following manner:

The Trap command contains an address in the user's program which, when encountered, will cause a transfer to SFCHEX. The Trap command also contains a parameter N, which indicates the number of times this transfer is to occur. (If N is input as a zero the transfer will occur for the remainder of the program run.) When the transfer occurs for the Nth time, the commands associated

with the Trap command are executed for the last time and SFCHEX returns for more input. Note that if N were input as a 1, the user could input more commands (e.g., parameter modifications by Correct command) as soon as the location included in the Trap command is reached.

When a Trap command is entered, all the subsequent commands are associated with that command until a Begin, End, Locate, Run, Wrapup or another Trap command is entered. This "association" is effected by storing the individual command parameters in a Trap Table. Snap and Unsnap commands in a Trap Table are treated differently than the other commands. The other commands are executed each time the Trap address is encountered. Whereas the first time that a Trap Table is executed, any Snap or Unsnap command will be set up, at the proper address in the user's program and then the command parameters will be obliterated in the table.

The length of a Trap Table is limited to forty-eight cells. The number of cells required for each allowable command is shown in the following list.

Command	Number	Explanation
Correct	4 + N	N is the number of cells to be corrected. $1 \geq N \leq 11$
Dump	5	
Heading	2 + N/2	N is the number of BCD characters
Page	1	
Snap	4	
Unsnap	5	
Mnemonic	5	

The Trap Table is said to have been "flushed" when the table has been executed for the Nth time. At this time, SFCHEX will perform a test to see if an End or Wrapup command has been entered. If neither has been entered, SFCHEX

will return for more commands. Otherwise the End command (2.2.2.4) procedure or the Wrapup command (2.2.2.12) procedure will be followed.

2.2.1.11 Unsnap Command

The Unsnap command is used to remove a Snap command which is currently in operation. When the address, included in the Unsnap command, is encountered in the user's program, the associated Snap command will be removed.

The address on the Unsnap command could be the same as the Snap command address. When the user's program encounters that address, the Snap command will be deleted. No Snap dump will be generated at this time.

2.2.1.12 Wrapup Command

This is a counterpart of the End command. It also indicates to SFCHEX that no more commands will be entered. However, in this case, when both the Trap Tables are flushed (except those which are indefinite, i.e., N input as 0), SFCHEX will terminate the run by halting. (If a monitor system is installed in the computer system, SFCHEX can be modified to return control to the monitor at this time.)

2.2.1.13 Mnemonic Dump Command

This command will cause 60 word blocks of core to be printed on the 166 printer. No conversion of data is done. No headers or address will be printed.

2.3 Command Formats

Figure 2.1 contains a column by column illustration of each of the command formats. The formats are identical for both card and typewriter input. SFCHEX allows recovery from any error resulting from command inputs in either input mode.

The following section is a description of the commands in relation to the formats presented in Figure 2.1. Notice that every command begins in column 1 and every command is terminated with a period.

2.3.1 Command Formats

2.3.1.1 Begin Command

FORMAT: B bXXXX.

EVENT: SFCHEX terminates the current input sequence and transfers program control to location XXXX in bank b.
Important: This command must be used to terminate the initial input command sequence.

2.3.1.2 Correct Command

FORMAT: C bXXXX $C_1C_1C_1C_1$ $C_2C_2C_2C_2$ $C_3C_3C_3C_3\cdots$

EVENT: $C_1C_1C_1C_1$ is stored in location XXXX, bank b.
$C_2C_2C_2C_2$ is stored in location XXXX + 1, bank b.
⋮
$C_nC_nC_nC_n$ is stored in location XXXX + (n-1), bank b.
(where $1 \leq n \leq 11$)

Cols.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	
	B			b	X	X	X	X	.																		
	C			b	X	X	X	X			C_1	C_1	C_1	C_1			C_2	C_2	C_2	C_2			C_3	C_3	C_3	C_3	...
	D			b_1	X	X	X	X			b_2	Y	Y	Y	.												
	E.																										
	H			C	O	M	M	E	N	T		U	P		T	O			C	O	L	7	9	.			
	L			b	X	X	X	X	.																		
	P.																										
	R.																										
	S			b	X	X	X	X			B	.															
	T			b	X	X	X	X			n	n	n	n	.												
	U			b	X	X	X	X			b	Y	Y	Y	Y	.											
	W .																										
	N			b	X	X	X	X			0	n	n	n	n	.											

Figure 2.1 COMMAND FORMATS

2.3.1.3 Dump Command

FORMAT: D b_1XXXX b_2YYYY.

EVENT: An octal dump from location XXXX bank b_1 through location YYYY bank b_2 will be generated on the on-line printer or read out on magnetic tape according to the user's option.

2.3.1.4 End Command

FORMAT: E.

EVENT: A switch is set so that SFCHEX will not attempt to read any more commands. Thus when all the prior commands have been completely executed the user's program will be allowed to continue until it terminates itself. After setting the above switch SFCHEX returns program control to the proper location in the user's program.

2.3.1.5 Header Command

FORMAT: H BCD COMMENT OF UP TO 76 CHARACTERS.

EVENT: The BCD command is generated on the on-line printer or magnetic tape depending on the user's option. The period may not be used as a part of the comment. If a period is inadvertently inserted in a comment, only those characters prior to that period will appear in the output.

2.3.1.6 Locate Command

FORMAT: L bXXXX.

EVENT: The nine cell RECOV Subroutine will be transferred from SFCHEX to locations XXXX through XXXX + 8 in bank b (the bank in which the user's program is stored).

Important: This command must be entered prior to any Trap, Snap or Unsnap command.

2.3.1.7 Page Command

FORMAT: P.

EVENT: A page will be ejected on the on-line printer or on the magnetic tape listing according to the user's option.

2.3.1.8 Run Command

FORMAT: R.

EVENT: SFCHEX will stop accepting input, terminate any Trap Tables that are being constructed and return to the user's program from which it last exited.

2.3.1.9 Snap Command

FORMAT: S bXXXX B.

EVENT: SFCHEX will place a JPR RECOV in locations XXXX through XXXX + 1 in bank b. Whenever the computer encounters cell XXXX in bank b control will be transferred to SFCHEX and a dump of cells $0\text{-}77_8$ in bank B along with the contents of the A Register will be made. Buffer Entrance Register and the bank settings will be output on the proper medium. If B was input as "4" a Snap dump of all four banks will occur.

2.3.1.10 Trap Command

FORMAT: T bXXXX nnnn.

EVENT: SFCHEX will place a JPR RECOV in locations XXXX and XXXX + 1 in bank b. One of the two TRAP TABLES will be selected and nnnn will be stored in the first cell of that table. Each time the computer encounters location XXXX in bank b the command parameters in the TRAP TABLE will be executed and nnnn will be decreased by 1. When nn is decreased to

zero the Trap command is removed and SFCHEX will accept input if any is forthcoming. (If nnnn was input as "0", a -1 is stored in the first cell of the TRAP TABLE and the Trap command is never removed.)

Subsequent commands are stored in the TRAP TABLE in the following manner. (Let TRAPSTOR be the next available location in the TRAP TABLE).

a. Correct Command in a TRAP TABLE

Command Format: C bXXXX $C_1C_1C_1C_1$ $C_2C_2C_2C_2$ . . . . .

Command Storage: The address of the CORRECT Subroutine is stored in TRPSTOR. b is stored in TRPSTOR + 1. XXXX is stored in TRPSTOR + 2. The number of corrections included in this command (i.e., n) is stored in TRPSTOR + 3.

$C_1C_1C_1C_1$ is stored in TRPSTOR + 4.

$C_2C_2C_2C_2$ is stored in TRPSTOR + 5.

. . . .

. . . .

. . . .

$C_nC_nC_nC_n$ is stored in TRPSTOR + (n + 3)

b. Dump Command in a TRAP TABLE

Command Format: D b_1XXXX b_2YYYY.

Command Storage: The address of the DUMP Subroutine is stored in TRPSTOR. b_1 is stored in TRPSTOR + 1. XXXX is stored in TRPSTOR + 2. b_2 is stored in TRPSTOR + 3. YYYY is stored in TRPSTOR + 4.

c. Header Command in a TRAP TABLE

Command Format: H H_1H_2 . . . . . H_n.

Command Storage: The address of the HEAD Subroutine is stored in TRPSTOR. Half the number of BCD characters (i.e., n/2) are stored in TRPSTOR + 1. H_1H_2 is stored in TRPSTOR + 2. H_3H_4 is stored in TRPSTOR + 3. . . . $H_{n-1}H_n$ is stored in TRPSTOR + (2 + n/2).

d. Page Command in a TRAP TABLE

Command Format: P.

Command Storage: The address of the PAGE Subroutine is stored in TRPSTOR.

e. Snap Command in a TRAP TABLE

Command Format: S bXXXX B.

Command Storage: The address of the Snap SETUP Subroutine is stored in TRPSTOR. b is stored in TRPSTOR + 1. XXXX is stored in TRPSTOR + 2. B is stored in TRPSTOR + 3.

f. Unsnap Command in a TRAP TABLE

Command Format: U b_1XXXX b_2YYYY.

Command Storage: The address of the Unsnap SETUP Subroutine is stored in TRPSTOR. b_1 is stored in TRPSTOR + 1. XXXX is stored in TRPSTOR + 2. b_2 is stored in TRPSTOR + 3. YYYY is stored in TRPSTOR + 4.

g. Mnemonic command in a TRAP TABLE

Command Format: N bXXXX ONNNN

Command Storage: The address of the mnemonic Subroutine is stored in TRPSTOR. b is stored in TRPSTOR + 1. XXXX is stored in TRPSTOR + 2. O is stored in TRPSTOR + 3. NNNN is stored in TRPSTOR + 4.

h. Begin, Run, End, Wrapup or a second Trap command will indicate to SFCHEX that the current TRAP TABLE is complete. In this case 7777 will be placed in TRPSTOR. The 7777 will serve as a flag to terminate the execution of the command parameters in the TRAP TABLE.

2.3.1.11 Unsnap Command

FORMAT: U

EVENT: SFCHEX will place a JPR RECOV in locations XXXX and XXXX + 1 in bank b_1. When the computer encounters location XXXX control will be transferred to SFCHEX. At this time the Snap command jump (i.e., JPR RECOV) will be removed from location YYYY and YYYY+ 1 in bank b_2. At the same time the Unsnap command jump will be removed from locations XXXX and XXXX +1. Then control will be returned to the user's program at location b_1XXXX.

2.3.1.12 Wrapup Command

FORMAT: W.

EVENT: A switch is set so that SFCHEX will not attempt to read in any more commands. Then the TRAP TABLES are checked. If both tables are empty (indefinite traps count as being empty in this regard) SFCHEX will terminate the program.

Otherwise SFCHEX returns control to the proper location in the user's program. However, when all the finite (i.e., not indefinite) traps have been flushed, SFCHEX will terminate the program.

2.3.1.13 Mnemonic Command

FORMAT: N bXXXX ONNNN

EVENT: A dump of core from location XXXX in bank b in groups of 60 word blocks specified by NNNN will be printed on the 166 line printer. No conversion of data is done. No headers or address will be printed.

3.0 OUTPUT

3.1 Output Modes

SFCHEX allows two modes of output; the 166 on-line printer or a 163 magnetic tape unit. SFCHEX will request the output option by printing on the typewriter:

MODE OUT

The user will then indicate his option by typing a "P" for 166 on-line printer output or a "Mx" (x being a tape handler number) for magnetic tape output.

Since selecting either BCD or Binary mode causes both tape handlers on the same unit to be set in that mode, SFCHEX will restore the tape unit to its original mode before returning control to the user's program. This allows the user to output in a binary format on the same unit which SFCHEX uses for BCD formats, without having to reset the modes.

3.2 Output Formats

The only normal output from SFCHEX are BCD annotations and octal dumps. (The Snap Command merely defines special limits for a dump.) A heading will precede the dump. This heading will contain the B, D, I and R Bank settings, the last address (P) encountered in the user's program, the contents of the A-Register at the time address (P) was encountered and the contents of the Buffer Entrance Register at that time.

3.2.1 Dump Format

The Dump command defines a lower limit address (LL) and an upper limit address (UL), between which the octal dump is to be generated. For the sake of uniformity, SFCHEX reduces LL to the nearest multiple of 16_{10}. Similarly, UL is increased to the nearest multiple of 16_{10}. This allows the output of an integral set of (16 cell) lines with the initial address of the cell at the beginning of each line an integral multiple of 20_8.

For additional ease in reading, the 16_{10} octal cells listed on each line are gathered into four groups, with the separation between words within a group being less than the separation between two groups. Figure 3.1 is a sample of the dump format.

3.3 Error Messages and Recovery Procedures

SFCHEX will generate comments on the on-line typewriter in the event of certain malfunctions due to erroneous command input procedures.

3.3.1 Illegal Input Character

If SFCHEX encounters an illegal character in the interpretation of a command, three question marks (? ? ?) will appear on the on-line typewriter.

Recovery Procedure

Typewriter input mode: SFCHEX will cause a carriage return and type an *. The user may retype the entire command.

Card input mode: The card should be repunched and the user may restart the program with that card by setting SLJ2 and starting at location 2334 in the bank in which SFCHEX is stored.

3.3.2 Illegal Instruction

Some instructions in the 160A repertoire are not allowed as cells in which a Snap or Trap jump can be replaced, since SFCHEX is unable to translate those instructions. (See Restrictions 4.1). When SFCHEX encounters such an illegal instruction, the following comment will be generated on the typewriter.

IL INST AT BXXXX

B will be the bank and XXXX the location in which the non-translatable instruction was stored.

Recovery Procedure

Typewriter input mode: SFCHEX will remove the Trap or Snap Command and then cause a carriage return and type an asterisk. The user may then reenter the Trap or Snap command at a different address or type in some other command.

B = 0 D = 0 I = 0 R = 0 P = 7500 A = 0000 BER = 7776

00200	0000	0176	0234	1111	7105	6666	7777	5000	1234	2222	3333	6666	0000	1111	2222	3333
00220	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
00240	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
00260	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
00300	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
00320	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

.
.
.

Figure 3.1 Sample Dump

Card input mode: SFCHEX will remove the Trap or Snap Command and stop. The user should correct or remove the erroneous command card. The user may restart the program with the corrected card by setting SLJ 2 and starting at location 2200 in SFCHEX's bank.

3.3.3 Illegal Input Command

If a spurious command is input to SFCHEX, the following message will be generated on the typewriter.

IL COM

Recovery Procedure

Typewriter input mode: SFCHEX will cause a carriage return and type an asterisk. The user may retype the command.

Card input mode: The erroneous card should be repunched and the user may restart the program with that card by setting SLJ 2 and starting at location 2200 in SFCHEX's bank.

3.3.4 Three Simultaneous Traps, Snaps, or Unsnaps

If the user's input command sequence should be so set up as to result in an attempt to set up a third Trap, Snap, or Unsnap command, the following message will be generated on the typewriter.

THIRD COM

Recovery Procedure

Typewriter input mode: SFCHEX will remove the third Snap, Trap or Unsnap command, cause a carriage return and type an asterisk. The user may then enter another command.

Card input mode: SFCHEX will remove the third Snap, Trap, or Unsnap and stop. The user should modify the command sequence. The user may restart the program at that point by setting SLJ 2 and starting at location 2200 in SFCHEX's bank.

3.3.5 Unsnap Command Address Does Not Match Any Snap Command Address

If the user should enter an Unsnap command in which the address of the Snap command to be removed cannot be matched with the address of any Snap command which is currently in effect, the following message will be generated on the typewriter.

NO S AT BXXXX

Recovery Procedure

Typewriter input mode: SFCHEX will remove the Unsnap command, cause a carriage return and type an asterisk. At this point the correct Unsnap command may be entered followed by a Run command (2.3.1.8).

Card input mode: SFCHEX will remove the Unsnap command, type out the error message and halt. To recover, the user must punch on cards the correct Unsnap command and/or a Run command (2.3.1.8), put jump switch 2 up and effect a manual jump from the console to location 2200 in the bank in which SFCHEX is located.

3.3.6 TRAP TABLE Capacity Exceeded

If a user exceeds the capacity of a TRAP TABLE which is being set up, the following message will be generated on the typewriter.

T T OVRFLW

Important: SFCHEX removes all the commands which were placed in the TRAP TABLE including the Trap command itself.

Recovery Procedure

Typewriter input mode: SFCHEX will remove the whole TRAP TABLE, cause a carriage return and type an asterisk. The user should decide the manner in which he wants to reduce the length of the TRAP TABLE to less than forty-eight cells and then begin the input by retyping the Trap command.

Card input mode: SFCHEX will remove the whole TRAP TABLE and then stop. The user should modify the TRAP TABLE sequence and begin with the Trap command by setting SLJ 2 and starting at location 2200 in SFCHEX's bank.

4.0 OPERATING INSTRUCTIONS

4.1 Options

4.1.1 Input Option

The user may input commands via the typewriter or the on-line card reader. The user signifies his choice by responding to SFCHEX's question "MODE IN" with a "T" for typewriter or a "C" for on-line card reader.

4.1.2 Output Option

The user may choose between the on-line printer or magnetic tape as his output medium. The user signifies his choice by responding to SFCHEX's question "MODE OUT" with a "P" for on-line printer or a "Mx" (x = a tape handler number) for magnetic tape output.

4.1.3 SLJ Switch Options

If SFCHEX has been used prior to the current situation, the present user may use the same input-output media as the previous user by setting SLJ 1 and starting the program. This will result in SFCHEX skipping the "MODE IN", "MODE OUT" questions.

SLJ 2 causes SFCHEX to bypass initialization of the trap tables.

4.2 SFCHEX Operation

SFCHEX will normally be loaded in the upper half of bank 2, starting at location 2200.

a. Load the user's program in any bank.
b. If the input mode is to be by card, the command cards should be placed in the card reader.
c. Start at location 2200 in bank 2.
d. Respond to "MODE IN" by typing a "T" or "C" on the typewriter.
e. Respond to "MODE OUT" by typing a "P" or "Mx" (x = the tape handler number).

6. SFCHEX will now begin to enter commands.
   a. If the input mode is by card, SFCHEX will read the cards as it needs them.
   b. If the input mode is by typewriter, SFCHEX will signal the user with a carriage return and an * each time it is prepared to accept a command.

4.3 Operating Restrictions and Limitations

4.3.1 Illegal Instructions

In the setting up of a Snap, Trap, or Unsnap command, SFCHEX places a JPR to the RECOV Subroutine in the indicated address in the user's program. It is a restriction that the two cells in which the JPR instruction is placed must not be modified in any manner. Any instruction whose E, F, or G field is modified in any manner may not be selected by the user for a Trap, Snap or Unsnap command. This includes the following situation.

```
A1      STR     TR1
A2      LDC
A3              1111
```

Suppose the user input a Snap command to be set up at A1, and suppose the contents of A3 are modified by another portion of the program, SFCHEX would translate the instruction at A1 and in the translation of the instruction at A2, SFCHEX would pick up the contents of A3 and save it in bank 2. Thus although modification of the contents of A3 would not affect the JPR command in A1 and A2, the change in A3 would go unnoticed.

Similarly, the second cell in which the JPR to RECOV sequence is stored must not be the object of some transfer instruction. The reason for this is obvious. When the jump is effected, the user's program will execute the G field of the JPR instruction.

4.3.2 Non-Translatable Instructions

The following instructions are not allowed to occur in the cells selected by the user for a Trap or Snap command.

(Note: These are allright for Snap commands.)

1. ERR Error stop
2. HLT Halt
3. JPR Return jump
4. SLS Selective stop
5. SJS Selective stop and jump
6. ALL INPUT-OUTPUT INSTRUCTIONS.

The reason that these instructions are not allowed is that SFCHEX has no translation for them. Inclusion of one of the above commands will cause SFCHEX to print out:

IL INST AT BXXXX (See 3.3.2)

4.3.3 The Locate Command

The Locate Command must be entered prior to any Trap, Snap or Unsnap command. This is due to the fact that the latter three commands assume that a communication link with the user's program has already been established.

4.3.4 The Begin Command

The Begin command must be used to terminate the initial set of input commands. This is due to the fact that SFCHEX does not have any address (at this time) in the user's program to which it can transfer program control.

4.3.5 TRAP TABLE Limit

The parameters associated with a Trap command must not exceed forty-eight cells of the corresponding TRAP TABLE. If too many commands are entered into the TRAP TABLE, SFCHEX will print-out.

TT OVRFLW (see 3.3.6)

The following is a list of the cells required for each type of command which can be placed in a TRAP TABLE.

Snap	4 cells
Unsnap	5 cells
Correct	(4 + n) cells (n is the number of corrections)
Header	(2 + n/2) cells (n is the number of BCD characters)
Dump	5 cells
Page	1 cell
Mnemonic	5 cells

4.3.6 Input-Output

If the input media (i.e., the typewriter or card reader) is not ready, SFCHEX will wait until the equipment is ready. No comment will be generated.

Likewise, if the output media is not ready, SFCHEX will wait and no comment will be generated.

The user will be able to observe either of the above conditions, since no commands are accepted by SFCHEX until both the input and output mediums have been tested.

4.3.7 Trap, Snap and Unsnap Commands

No more than 2 each of the Snap, Trap or Unsnap commands can be active at the same time. By active, we mean that a JPR RECOV has been placed in the user's program. You could have a third Snap or Unsnap command in a TRAP TABLE which has not been set up. But when the Snap or Unsnap command did get set up, one of the other Snap or Unsnap commands will have had to be removed. A third active command will result in the error message:

THIRD COM. (See 3.3.4)

4.3.8 Pressing the Load Button On the 167-2 On-Line Card Reader

In validation, it was observed that the depression of the load button on the 167-2 on-line card reader resulted in an over-ride of the typewriter input option selection. The indication of such a mistake will be an error printout loop of the following message

IL INST AT 2020

4.3.9 Equipment Configuration

SFCHEX is a diagnostic program with storage requirements of 2850 cells. It is bank independent with the exception of storage in bank 3. For full capability, the equipment configuration must consist of a 160A/169 computer, a 161 typewriter, a 167-2 on-line card reader, a 166 on-line printer and a 163 tape unit.

The timing requirements for the operation of SFCHEX is a function of the amount of input commands.

5.0 SFCHEX Capability

5.1 The Trap Command

The Trap command format and operation is described in detail in Section 2. When a Trap command is entered, a flagged jump is placed in the user's program so that when that jump is encountered, control is transferred to SFCHEX via the RECOV Subroutine. At this time the Trap Table associated with the flagged jump is executed; that is, each command which was input in connection with the Trap command will either be set up or executed. Any Snap or Unsnap commands are set up by placing a flagged jump in the user's program at the indicated address. Correct commands are executed by storing the parameters in the proper locations. The Snap, Unsnap and Correct commands are then deleted from the Trap Table so that subsequent executions of the Trap Table will not result in redundant and perhaps disastrous reexecutions of the above commands. Dump, Header and Page commands will be executed each time the Trap Table is executed.

The key feature of the Trap command is the parameter n (the number of times the Trap Table is to be executed). If n is entered on the Trap command as 0, the associated Trap Table will continue to be executed each time the flagged jump is encountered. This will continue for the duration of the program. If n is input with a positive value, the Trap Table will be executed up to n times at which time the Trap Table will be "flushed". "Flushing" consists of the removing of the flagged jump in the user's program, replacing the instructions at that location and terminating the Trap Table. (If n is not given the Table shall be executed only once.)

At this time SFCHEX will be ready to accept more commands. The user may decline to enter more commands at this time by using the Run command, or he may decline to enter any more commands for the duration of the run by using an End or Wrapup command. Otherwise the user may enter any sequence of commands which is compatible with the restrictions and limitations listed in Section 4. However, if the user desires to communicate with SFCHEX at some later time, he

must make sure that a Trap Table will flush at that time. The Trap command can be used strictly for communication purposes, that is, it is not necessary to associate any commands with it (the Trap Table may be empty). Thus, each time the corresponding flagged jump is encountered in the user's program, n will be decreased by one. When n is reduced to zero, the user will be allowed to enter more commands.

5.2 SFCHEX Strategy

The following paragraphs contain examples of debugging situations for which SFCHEX is ideally suited, and advice for judicious use of the commands.

5.2.1 With two Trap Tables available, it would be unwise to set up an infinite trap (n input as 0), unless the user has only minimal debugging procedures planned for the run. Once an infinite Trap Table is set up, it cannot be removed until SFCHEX is restarted.

5.2.2 Consider a program which consists of several modules and which is not sufficiently checked out for a continuous run through the modules. Normally, the user program has to check out the modules one by one by separating the program and writing small drivers for the modules or by attending the computer runs and manually transferring from one module to the next. With SFCHEX, the user could set up a continuous run by discrete use of Trap, Correct, Dump and Begin commands. For example, set a Trap command at the end of the first module with n equal to one. Use the Begin command to start at the proper location in the first module. When the program encounters the Trap Flag, the user could execute a Dump of the first module, use the Correct command to set up ideal parameters for the next module, enter a Trap Command at the end of the next module and then use the Begin command to start at the proper location of the next module. This process can continue for each module until the run is completed.

5.2.3 Remember that the only commands which terminate a Trap Table are the Run, Begin, End, Wrapup or another Trap command. Therefore, in an input period,

if any Header, Page, Snap, Unsnap, Dump, or Correct commands are to be executed immediately, they must be input prior to any Trap Commands, since the commands which terminate the Trap Table also cause SFCHEX to transfer control to the user's program.

5.2.4 When a programmer desires to test the effect of several different values of some parameter, he could run a parameter study using Trap, Begin and Correct commands. The procedure would be to set a Trap command at the end of the program with n equal to 1 and use the Begin command to start the program. When the first run is complete, the Trap will allow the programmer to use the Correct command to enter a new value for the parameter, enter a new Trap Command with n equal to one and use the Begin command to restart the program.

6.0 TEST CASES

6.1 Procedures

In checking out SFCHEX the programmer followed the normal procedures for checking out the individual sections and modules for logical and diagnostic errors. Each of the input commands was individually checked to see if it functioned correctly. Several different computer instruction translations were run off by the SETUP Subroutine and checked against hand written results. As soon as SFCHEX began to run in normal fashion it was turned upon itself. That is, SFCHEX corrected, snapped, trapped, and dumped itself.

The validation specifications were essentially SFCHEX' only really complete test. The following pages of this section will deal with the validation specifications appearing in the Milestone VI Document.

6.2 Sample User's Program

A small program was generated to serve as a sample user's program on which SFCHEX could operate. This program contains a variety of instructions so that each of the nine instruction translation modules in the SETUP Subroutine can be exercised. The program was designed to iterate 4 times and then halt. Each iteration of the program is divided into two sections. The first section fills a 896 cell buffer with a series of 16 0000's, 16 1111's, 16 2222's, etc, until all 896 cells are filled. The second fills the same buffer with 7777's. The program was coded in a roundabout manner so that an adequate cross section of commands could be used. The following pages contain the symbolic coding of the sample user's program.

SAMPLE PROGRAM

LOC	TAG	OPER	ADD	SETUP MODULE
		ORG	100	
100		SICO		
101		SDCO		

LOC	TAG	OPER	ADD	SETUP MODULE
102		LDN	40	
103	STM	TST17		
104				
105	TST1	LDN	70	MOD1C
106		STD	TD1	MOD1C
107		LDN	0	MOD1C
110		STM	TST3	MOD1B
111				
112		LDC	TSTBF	MOD1D
113				
114		STD	TD2	
115	TST8	LDN	20	
116		STM	TST4	
117				
120	TST5	LDM	TST3	
121				
122		STI	TD2	
123		AOD	TD2	
124		LDM	TST4	
125				
126		SBN	1	MOD1C
127		STR	TST4	MOD1A
130		NZR	TST15	
131		LCN	1	
132		RAD	TD1	
133		ZJR	TST6	MOD2A
134		LDM	TST3	MOD1B
135				
136		ADC	1111	
137				
140		STR	TST3	

LOC	TAG	OPER	ADD	SETUP MODULE
141		LDN	1	
142		NZR	TST8	
143	TST6	PTA		MOD9
144		ADN	5	MOD1C
145		STD	TD3	MOD1C
146		JPI	TD3	MOD2B
147		NOP		
150		STPO		MOD8
151		LDN	4	MOD1C
152		RAD	50	MOD1C
153		JPI	50	MOD2B
154		SLJ	1	MOD4
155			TS15	
156	TST15	PTA		MOD9
157		ADN	3	MOD1C
160		SRJO		MOD5
161		LDC	TSTBF	
162				
163	TST12	ATE	TST12	MOD7
164				
165	TST13	LDC	TSTBX	
166				
167		ATX	TST13	MOD7
170				
171		SBUO		MOD6
172		LCN	1	MOD1C
173	TST14	BLS	TST14	MOD7
174				
175		LCN	1	MOD1C
176		RAR	TST17	MOD1A
177		ZJR	TST18	MOD2A

LOC	TAG	OPER	ADD	SETUP MODULE
200		JFI	1	MOD3
201			TST1	
202	TST18	HLT		ILLEGAL
203	TST17			
204	TST3			
205	TST4			
		CON	1	
1	TD1			
2	TD2			
3	TD3			
		ORG	300	
300	TSTBF	BSS	896D	
2100	TSTBX			
		END		

6.3 Validation Prodecures

SFCHEX was validated and accepted on 21 December 1962. The validation tests concentrated on the following main points.

a. The different SFCHEX commands were entered in various sequences, executed and the resulting output corresponded to specified results.

b. Several different types of 160A computer instructions in the sample user's program were translated by the SETUP Subroutine in SFCHEX. The continued accurate running of the sample user's program was taken as an adequate indication that the instructions were being translated correctly.

c. Both the typewriter and card input modes and both the printer and magnetic tape output modes were exercised in all four possible input-output combinations. Naturally, not all of the possible

command sequences could be tried, but this caused no problems due to the fact that the command procedures are autonomous (except for the Trap Table Sequences).

The five following tests were designed to validate the command execution, the table capacities and the diagnostic stops in SFCHEX. They are reproduced in the same format as presented in the Milestone VI Document.

6.3.1 Test One

This test will exercise all of the translation modules in the SETUP Subroutine, and also all of the command types.

6.3.1.1 Command Card Sequence

COM	COL. 4	INST. MODE TRANSLATED
C	00107 0400 4100 0204.	
L	03000.	
C	00100 0020.	
P		
H	TEST ONE.	
H	INITIAL DUMP.	
D	00100 00202.	
T	00105 0002.	MOD1B MOD1C
S	00112 0	MOD1D
H	TRAP DUMP.	
D	00300 00340.	
B	00100.	
S	00146 0.	MOD2B
U	00107 00112.	
U	00150 00046.	
T	00154 0002.	MOD4
T	00160 0002.	MOD5

COM	COL. 4		INST. MODE TRANSLATED
S	00160	0002.	MOD5
R.			
U	00163	00150.	
R.			
T	00171	0002.	MOD6
T	00200	0006.	MOD3
C	00004	1234.	
S	00163	0.	MOD7
R.			
S	00176	0.	MOD1A MOD2A
U	00150	00163.	
U	00107	00176.	
R.			
T	00143	0002.	MOD9
W.			

6.3.1.2 Procedure

a. Read in SFCHEX Program.
b. Read SAMPLE User's Program into bank 0.
c. Place Command Cards in Card Reader.
d. Start at initial instruction of SFCHEX.
e. Respond to MODE IN with a C on the typewriter.
f. Respond to MODE OUT with a P on the typewriter.

6.3.1.3 Acceptance Criteria

a. Output will appear on the on-line printer.
b. Cards will be read in from the on-line card reader.
c. Output Sequence

1. "TEST ONE"
2. "INITIAL DUMP"
3. Dump from 00100 to 00202
4. "TRAP DUMP"
5. Dump from 00300 to 00740
6. Dump 0-77 bank 0
   Cell 1 0070
   Cell 2 0000
   Cell 3 0000
   Cell 50 0000
7. Same as d.
8. Dump from 00300 to 00340 containing 7777's
9. Dump 0-77 bank 0
   Cell 1 0
   Cell 2 2100
   Cell 3 150
10. Same as i
11. Dump 0-77 bank 0
    Cell 1 0000
    Cell 2 2100
    Cell 3 0150
    Cell 4 1234
12. Same as k
13. Program will halt in SFCHEX program.

6.3.2 Test Two

This test is the same as Test One with Step A of the procedure (6.3.1.2) deleted.

6.3.2.1 Command Card Sequence

Same as in 6.3.1.1

6.3.2.2 Procedure

a. Read in SAMPLE User's program.
b. Place Command Cards in Card Reader.
c. Start at initial instruction of SFCHEX with JS 1 set if same input/output modes (therefore skip steps D and E which follow).
d. Respond to MODE IN with a C on the typewriter.
e. Respond to MODE OUT with a P on the typewriter.

6.3.2.3 Acceptance Criteria

The results should be identical to the criteria in 6.3.1.3. This will indicate that SFCHEX can be used by successive programs.

6.3.3 Test Three

This test is similar to Test Two with Step A of the procedure deleted and input by typewriter rather than card.

6.3.3.1 Command Sequence

Type the same command sequence as given in 6.3.1.1.

6.3.3.2 Procedure

a. Start at initial instruction in SFCHEX with JS 1 down.
b. Respond to MODE IN with a T on the typewriter.
c. Respond to MODE OUT with a P on the typewriter.
d. Each time the typewriter does a carriage return followed by an asterisk type in the next command.

   Hit the carriage return before typing the period and re-enter the command. This will indicate that erroneous instructions may be changed.

6.3.3.3 Acceptance Criteria

The results should be identical to the criteria in 6.3.1.3. This

will indicate that the typewriter mode and card mode give identical results and also that SFCHEX does not disturb the user's program. (Unless an indefinite trap were placed in the program.)

6.3.4 Test Four

This test will use two T, two S and two U commands simultaneously to validate the table lengths.

6.3.4.1 Command Card Sequence

COM	COL. 4		INST. MODE TRANSLATED	
L	03000			
H	TEST FOUR.			
T	00105	0002.	MOD1C	MOD1C
S	00107	0.	MOD1C	MOD1B
D	00000	00017.		
T	00143	00020.	MOD9	MOD1C
S	00145	4.	MOD1C	MOD2B
D	00020	00037.		
B	00100.			
T	00177	0000.	MOD2A	MOD3
U	00107	00107.		
U	00145	00145.		
R.				
T	00105	0.	MOD1C	MOD1C
H	FINAL	DUMP.		
D	00000	37777.		
E.				

6.3.4.2 Procedure

a. Read in SFCHEX program

b. Read in SAMPLE user's program

c. Place Command Cards in Card Reader
d. Start at initial instruction of SFCHEX
e. Respond to MODE IN with a C on the typewriter
f. Respond to MODE OUT with a Mx (where x is an available magnetic tape unit).

6.3.4.3 Acceptance Criteria

a. Output will be on magnetic tape
b. Cards will read in from the on-line card reader
c. Program will terminate at 0202 in the user's program
d. List Magnetic Tape out will be
e. "TEST FOUR"
f. Dump 0-17 bank 0
g. Dump 0-17 bank 0
h. Dump 20-37 bank 0
i. Dump 0-77 bank 0
Dump 0-77 bank 1
Dump 0-77 bank 2
Dump 0-77 bank 3
j. Same as f
k. Same as h
l. "FINAL DUMP"
m. Dump of core memory

6.3.5 Test Five

This test series will demonstrate the error indications. All input will be typed.

6.3.5.1 Procedure

a. Load SFCHEX
b. Load SAMPLE user's program into bank 0.
c. Start at initial cell in SFCHEX

d. Respond to MODE In with an X

SFCHEX will retype MODE IN

e. Respond with a T

f. Respond to MODE OUT with a P

g. Type the command X 00100.

SFCHEX will type IL COM and request another command.

h. Type L 03000.

H TEST.

"TEST" will appear on the on-line printer

i. Type T 0202.

This is an illegal format. SFCHEX will request another character.

j. Start again at initial cell in SFCHEX by using B command

k. Respond to MODE IN with a T

l. Respond to MODE OUT with Mx (where x is an available magnetic tape).

m. Type the following commands:

```
L   03000.
T   00105  0000.
T   00107  0000.
T   00122  0000.
```

SFCHEX will type out "THIRD COM"

n. Repeat J through L

o. Type the following commands

```
C   00105  0470  4001  0400  4100
S   00105  0.
U   00107  00107
B   00105.
```

1. SFCHEX will output snap on tape x
2. Since Unsnap command does not agree with snap command address, SFCHEX will type out:

"NO S AT 00107"

p. Type the following commands:

```
C   00105  0470  4001  0400 4100.
T   00105  0000.
H   0102030405060708091011121314.
H   0102030405060708091011121314.
H   0102030405060708091011121314.
```

At this point 49 cells of the TRAP TABLE are filled.

Type H 01.

SFCHEX will type out "TT OVRFLW", the typewriter will do a carriage return and type an *.

At this point the trap set up at 00105 has been replaced with the original instructions and the trap has been removed from the TAB table. Thus this location may be trapped again.

The test is completed.

7.0 PROGRAM DESCRIPTION

This section presents both a verbal and a drawn flowchart breakdown of the submodules of SFCHEX.

The program naturally divides into two sections. The first section deals with the input, setup and execution of the commands. The second section is entered whenever a Snap, Trap or Unsnap command causes a transfer from the user's program to SFCHEX.

7.1 General Considerations

The subroutine and section descriptions on the following pages concentrate principally on the logical processes in the program. Each subroutine will be presented in sufficient detail so that any individual who is investigating a listing of SFCHEX can correlate the listing with the flowcharts. For this purpose, the flowcharts will use the same tags as appear in the listing, rather than mnemonic tags which would have more contextual meaning.

Flow Chart Conventions

(X)	Contents of X
[(X)]	Contents of address specified by contents of X
$\longrightarrow$	Assignment (e.g. 1 $\longrightarrow$(X))
$\odot$	Logical product

7.1.1 Tables and Buffers

The various storage tables and buffers used in SFCHEX will be explained in this section so that their use in the flowcharts will not necessitate redundant explanations.

7.1.1.1 BUF Buffer (41 cells)

The BUF Buffer is used to store the parameters from the current input command. If the command is to be executed or set up immediately, the data will be taken directly from BUF. If the command parameters are to be saved in a Trap Table, the parameters are transferred from BUF to one of the two Trap Tables (TT Buffer).

7.1.1.1.1 BBUF1 and BBUF2, the NEED Buffers (64 cells apiece)

These two buffers are stored in the upper portion of bank 3. They are used by the NEED and UNNEED Subroutines for storing the contents of the direct cells used by SFCHEX and the user's program. This allows SFCHEX to "borrow" the direct cell locations for more efficient operation and then restore the direct cells when control is returned to the user's program.

7.1.1.2 OUTBUF Buffer (56 cells)

This buffer is used to set up the dump output messages.

7.1.1.3 SNTB Table (2 cells)

This table contains the banks which are to be "Snap" dumped. The proper bank in SNTB will be selected according to the original position of the flagged address in the TAB Table.

7.1.1.4 SX Table (60 cells)

SX is the Snap Instruction Execution Table. It is divided into two 30 cell sections corresponding to the two possible Snap commands. This buffer contains the translations of the two instructions which were removed from the user's program when a Snap command was setup. Thus, when the flagged address in the user's program is encountered, SFCHEX will generate the Snap dump and then transfer to the proper half of the SX Table and execute the translations contained therein.

7.1.1.5 TAB Table (18 cells)

When a Snap, Trap or Unsnap command is set up, SFCHEX stores a JPR to the RECOV Subroutine in the user's program at the required address. This means that two instructions in the user's program must be saved in SFCHEX until that command is terminated.

TAB is divided into three 6 cell sections. The first 6 cells will contain the user's program address and the two saved instructions for the two Trap Commands. The second set of 6 cells will contain the addresses and instructions for the two Unsnap commands. The final 6 cell sections will contain

the address and instructions for the two Snap commands. As stated above, each 6 cell section is divided into two 3 cell subsections. The first cell of a subsection will contain the address in the user's program from which the instructions were taken. The second cell will contain the instruction taken from that address. The third cell will contain the instruction taken from the location immediately following that address.

TAB serves the additional function of bookkeeper for the SX, TX, UNSTBL and SNTB Tables. Whenever a flagged jump in the user's program is encountered, the address from which the jump was effected will be compared with the addresses in the TAB Table. Counters are set so that when the proper address is found, SFCHEX will be able to pick up the corresponding elements in the TX, SX, UNSTBL and SNTB Tables.

In order to keep track of the available storage in the TAB Buffer, a 7777 is stored in the initial cell of each 3 cell subsection which is currently not in use.

7.1.1.6 TT Buffer (100 cells)

This buffer contains the two Trap Tables. The Trap Tables are set up so that the number n (number of times the trap is to be executed) is contained in the first cell of the table. As subsequent commands are processed, the address of the corresponding command execution subroutine and the pertinent parameters will be stored in the table. When another Trap command or a Begin, Run, End or Wrapup command is read, a 777 is stored after the last parameter in the Trap Table.

Thus, when a Trap Table is executed, the n in the initial cell will be reduced by one. Then a transfer will be made to the first command execution address in the Trap Table. The command execution subroutine will pick up its parameters from the Trap Table. When the command function is completed, it will return control in such a manner that the address of the next subroutine to be executed from the Trap Table will be readily available. When the 7777

at the end of the Trap Table is sensed, the Trap Table execution is completed.

7.1.1.7 TX Table (60 cells)

This is the counterpart of the SX Table for the Snap command. TX is the Trap Instruction Execution Table. It is divided into two 30 cell sections corresponding to the two possible Trap commands. This buffer contains the translations of the two instructions which were removed from the user's program when a Trap command is set up. Thus,when the flagged address in the user's program is encountered, SFCHEX will execute the corresponding Trap Table and the transfer to the proper half of the TX Table and execute the translations contained therein.

7.1.1.8 TYPBCD Table (64 cells)

This is the typewriter BCD code conversion table.

7.1.1.9 UNSTBL Table (2 cells)

This table contains the addresses of Snap commands which are to be removed when the flagged jump for the corresponding Unsnap command is encountered.

7.2 Input, Setup and Exeuction Section (Figure 7.1)

Figure 7.1 shows the initial breakdown of this section.

The procedure is:

a. Upon entry, if jump switch 2 is set, SFCHEX will transfer immediately to the NEED Subroutine (step e). This switch is used only when there has been a card input error.

b. Certain switches and buffers are initialized. Since SFCHEX is bank independent, the relative bank for SFCHEX is determined and stored.

c. If jump switch 1 is set, the program will skip step d. This allows the user to keep the same input and output modes as were used for the previous run.

d. Next, the input-output options are set up (Section 7.2.1).

e. The NEED Subroutine is called (Section 7.2.2). This subroutine saves the contents of the user's direct cells, and stored SFCHEX parameters in the direct cells.

f. Finally, the command input section (Section 7.2.3 - 7.2.3.13) is entered and the initial commands are entered and set up or executed. When all the initial commands have been entered, SFCHEX will transfer control to the bank and address indicated by the Begin command.

7.2.1 I-O Section (Figure 7.2)

This section establishes the input and output options and verifies equipment status for the desired modes.

The procedure is:

a. "MODE IN" is written on the on-line typewriter. The answering T or C is stored in Input.

b. If C is the indicated option, the card reader status is tested until the card reader is ready.

c. "MODE OUT" is then written on the on-line typewriter. The answering P or MX is stored in OUT 2.

d. The output equipment status is tested until the equipment is ready.

e. Go to BETA, the tag for the jump to the NEED Subroutine (Figure 1).

7.2.2 NEED Subroutine (Figure 7.3)

The NEED Subroutine is used to save the direct cells in the user's relative bank so that SFCHEX may use those cells for storage and transferral of data.

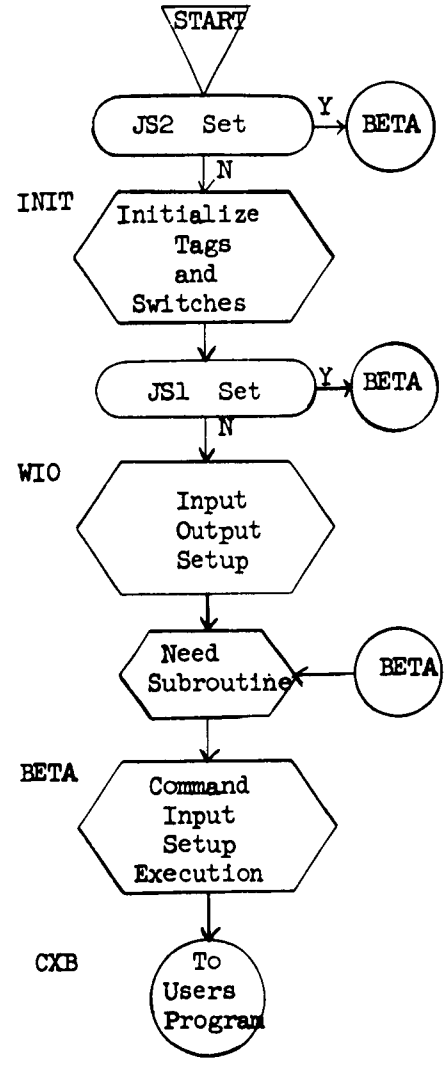

Figure 7.1 Input, Setup and Execution

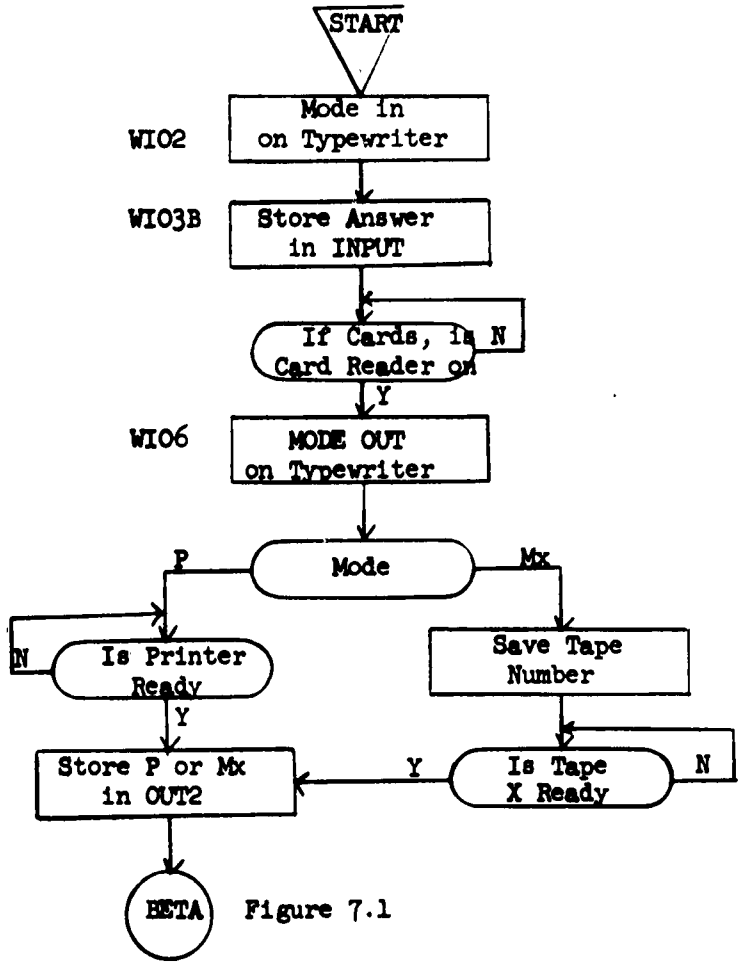

Figure 7.1

Figure 7.2 Input-Output Section

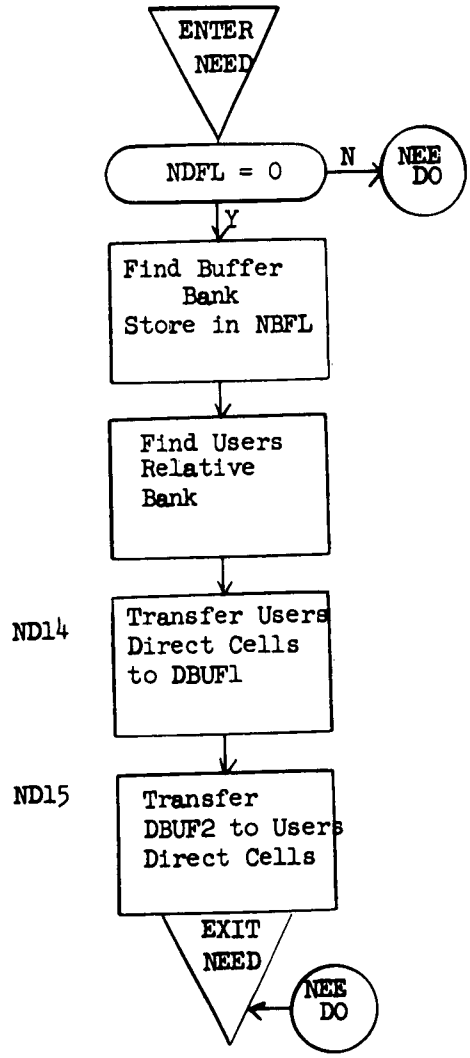

Figure 7.3 NEED Subroutine

The procedure is:

a. Test NDFL. 0 - go to step b. If NDFL does not equal zero, then the information has already been transferred. To to step 6.

b. Find the buffer bank and store it in NDFL.

c. Find the user's relative bank.

d. Transfer the direct cells in the user's relative bank to DBUF1.

e. Transfer the contents of DBUF2 into the direct cells of the user's relative bank.

7.2.3 Command Input (Figure 7.4)

Figure 7.4 is schematic representation of the multiple switch involved in transferring to the proper command interpretation Subroutine.

Thus the command is read in BCD format into the BUF Buffer. As shown in the Input Section (Section 2) the mnemonic letter representing the command is always in column one, and is therefore read into the first cell of BUF. The contents of this cell are compared with stored BCD characters and the corresponding subroutine is entered. If no character match is found the error subroutine EX2 is executed,resulting in the following printout on the on-line typewriter.

"IL COM"

The following sections (7.2.3.2 thru 7.2.3.13) deal with the processing of the individual commands.

7.2.3.1 BUF Buffer, Trap Table and SW1

The BUF BUFFER contains the BCD character as read in through the input medium. The individual subroutines will convert the BCD characters into the corresponding parameters and restore them in the BUF Buffer in the correct parameter sequence. To save instructions, the BUF Buffer is located in the direct cells of the user's relative bank.

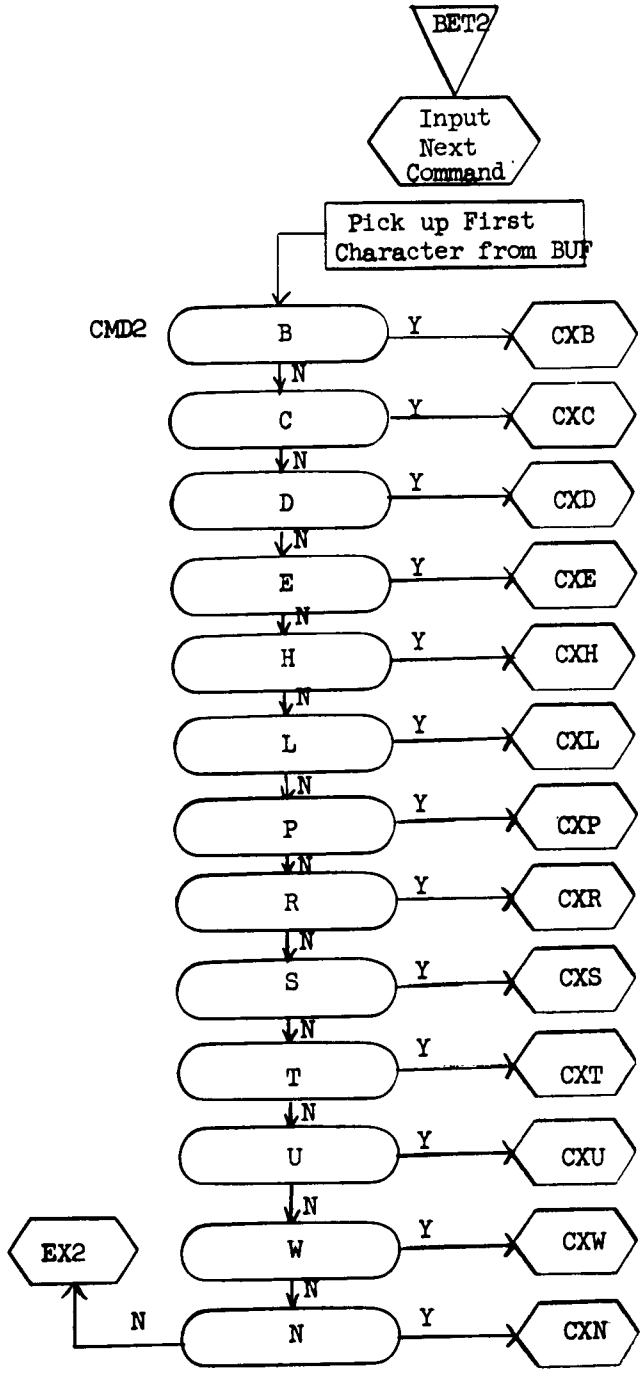

Figure 7.4 Command Test Sequence

SW1 is the switch used for keeping track of whether the next input command is to be stored in the Trap Table or if it is to be immediately set up or executed. If SW1 equals 1, the former situation is indicated and if SW1 equals 0, the latter situation is indicated. (See Sections 7.1.3.2 thru 7.2.3.13).

In the case where SW1 equals 1, the converted parameters which would normally be restored in the BUF Buffer are stored instead in the current Trap Table. The next available address in the Trap Table is contained in the direct cell labeled CD1.

7.2.3.2 CXB Begin, Command - (Figure 7.5)

The Begin Command is used to terminate the initial sequence of commands and to indicate the address and bank to which SFCHEX is to transfer control. (This address will normally be the location of the initial instruction of the user's program.)

When CXB is entered, SW1 is initially tested. SW1 will equal 1 if the last command was inserted in a Trap Table. SW1 will equal 0 if the last command was not associated with a TRAP Table.

The procedure followed is:

a. Test SW1. 0 - go to step d. 1 - continue.

b. Store 7777 in the next available location in the Trap Table (via CD1, which contains the address of that cell). This will terminate the Trap Table.

c. Set SW1 to 0. This will initialize the input section for the next input command sequence.

d. Call XADD Subroutine. This will cause the characters in BUF+1 through BUF+3 to be converted to the bank number and the address which will be stored in BUF+1 and BUF+2 respectively.

e. Call UNNEED Subroutine. This will transfer the parameters in the direct cells of the user's relative bank to DBUF2. Then the parameters in DBUF1 will be transferred to the direct cells in the user's relative bank.

f. Transfer to the address and bank stored in BUF+2 and BUF+1 respectively.

7.2.3.3 CXC Correct Command (Figure 7.6)

The Correct Command enters up to 11 corrections sequentially starting at the indicated address and bank.

The procedure is:

a. Test SW1. 0 - go to step q. 1 - continue.

b. The address of the CORRECT Subroutine is placed in the next available location in the TRAP Table using CD1. Thus, when the TRAP Table is executed for the first time, CORRECT will be called and the corrections will be transferred from the TRAP Table to the desired location.

c. Increase CD1 by one and call OVF Subroutine. OVF merely checks to see if a Trap Table overflow has occurred. An overflow will cause the Trap Table to be deleted and transfer to BET2. (Figure 7.4).

d. Call XADD. This subroutine picks up the contents of BUF+1 thru BUF+3 and stores the resulting bank and address via CD1.

e. The contents of CD1 is increased by 2 and stored in DTR. (The number of corrections (CD3) will be stored in the address stored in DTR.)

f. Increase CD1 by 1.

g. Set CD2 to address in BUF where the first correction parameters are stored.

h. Does (CD2) contain a period? Yes - go to step n, no - continue.

i. Use BINC Subroutine to convert the BCD characters in (CD2) and (CD2+1) to a binary parameter.

j. Store the result in the Trap Table via CD1. Increase CD1 by 1.

k. Call OVF Subroutine. If the Trap Table has overflowed, the table will be removed. TT OVRFLW will appear on the typewriter and SFCHEX will transfer to BET2 or stop depending on the input option.

l. Increase CD2 by 2, increase CD3 by 1.

m. Go to step h.

n. At this point, all the corrections have been stored. CD3 contains the number of corrections and is stored in the location whose address was saved in DTR.

o. Increase CD1 by one.

p. Go back to BET2 and input the next command.

q. Since SW1 was set to 0, the correction will be executed immediately.

r. Steps d thru m are executed with the appropriate BUF address inserted in CD1. This results in the bank and address stored in BUF+1 and BUF+2, the number of corrections in BUF+3, and the N corrections stored in BUF+4 thru BUF+N+3.

s. Call CORRECT Subroutine (Section 7.3.1) with the address of BUF in CD1. CORRECT will store the corrections.

t. Go back to BET2 and input the next command.

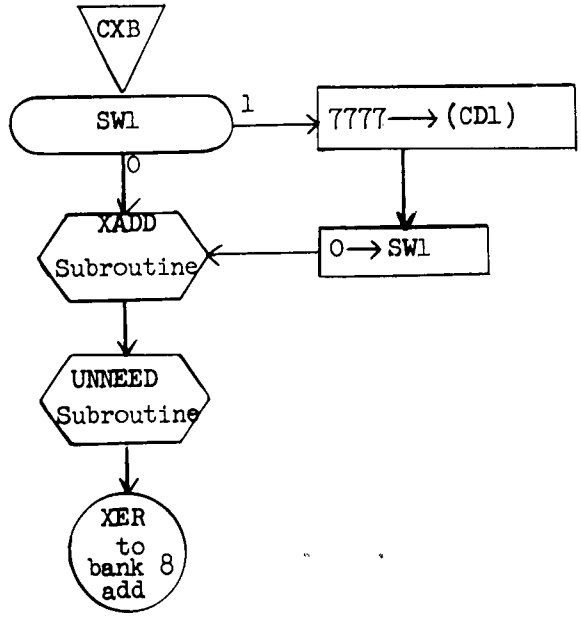

Figure 7.5 CXB

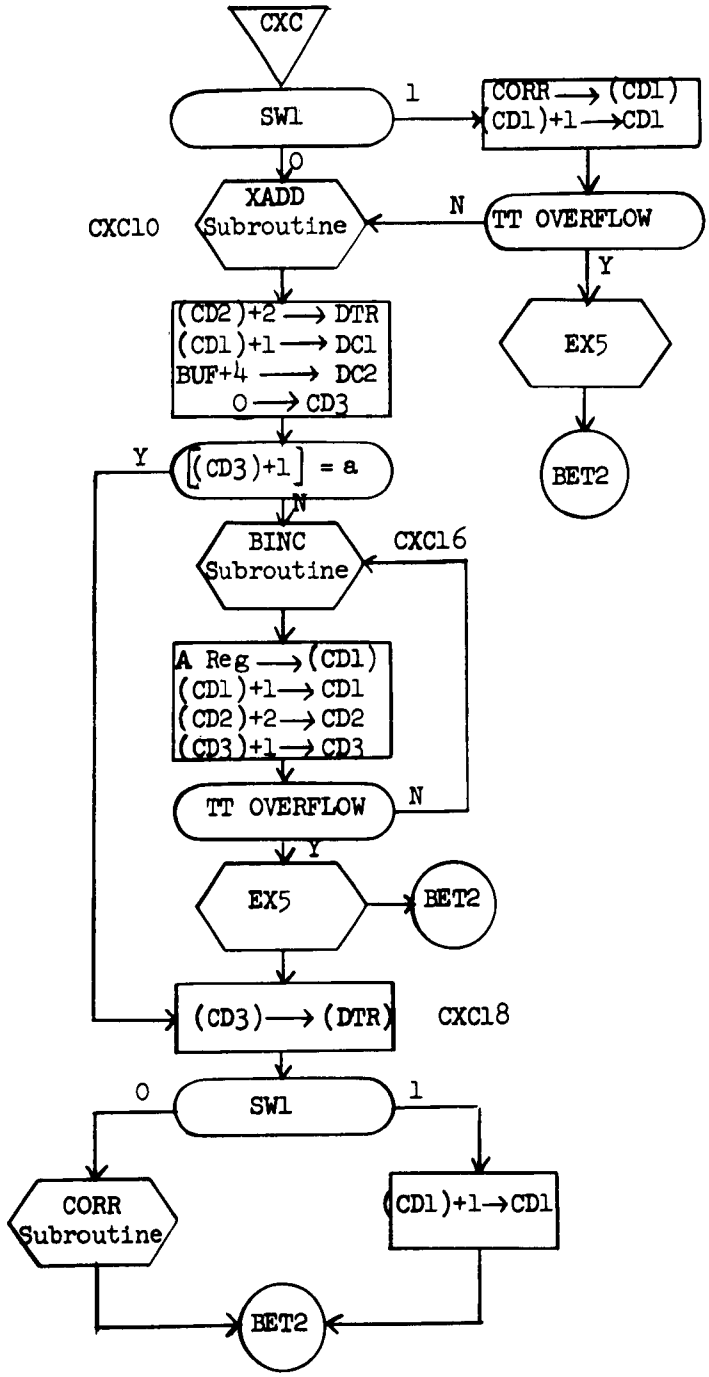

Figure 7.6 CXC

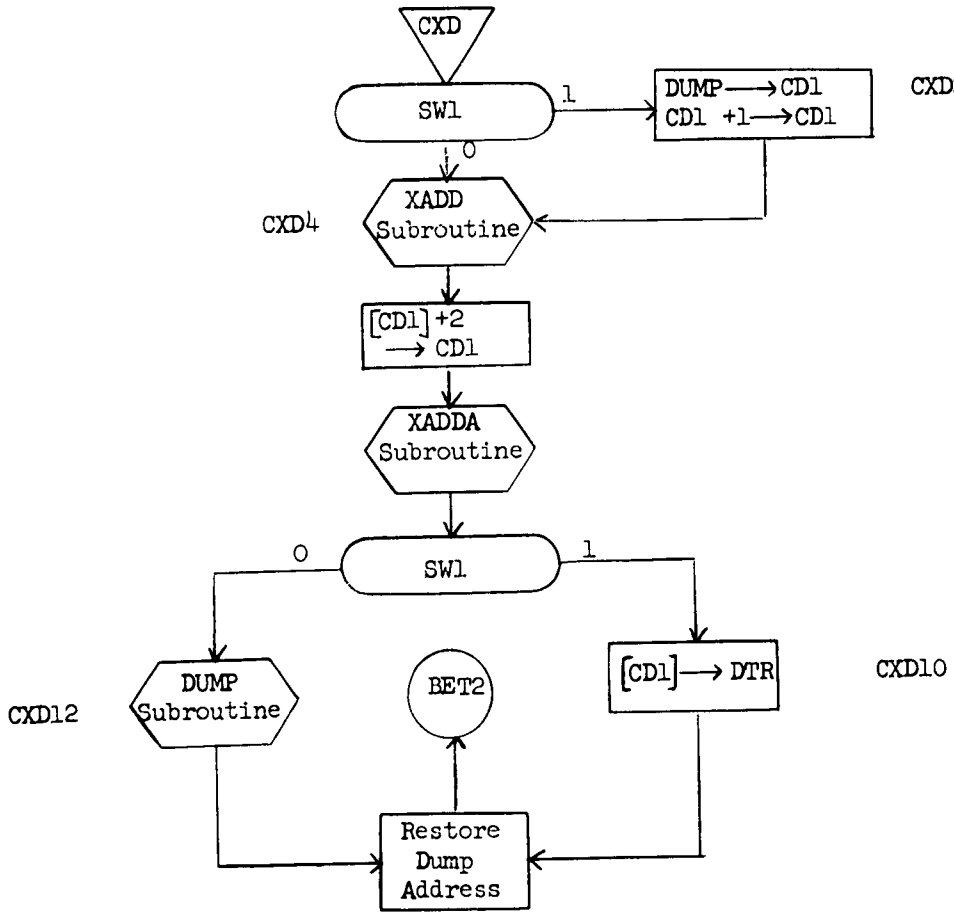

Figure 7.7 CXD

7.2.3.4 CXD Dump Command (Figure 7.7)

This command causes a dump of specified limits to be effected.

The procedure is:

a. Test SW1. 0 - go to step i. 1 - continue.

b. Store address of DUMP Subroutine in the TRAP Table via CD1.

c. Increase CD1 by one.

d. Use XADD to get lower limit of dump. (This will be stored in the TRAP Table via CD1.)

e. Increase CD1 by 2.

f. Use XADDA to convert the contents of BUF+4 thru BUF+6 to the upper limit of the dump. XADDA is the counterpart of XADD, only it deals with the contents of BUF+4 thru BUF+6.

g. Store the contents of CD1 in DJR.

h. Go to BET2 and input the next command.

i. Perform steps c thru h with the appropriate BUF address in CD1.

j. Call DUMP Subroutine (Section 7.3.2).

k. Go to BET2 and input the next command.

7.2.3.5 CXE End Command (Figure 7.8)

The command is used to indicate that no more commands are to be input.

The procedure is:

a. Test SW1. 0 - go to step d. 1 - continue.

b. Terminate TRAP Table with a 7777. (Stored through CD1).

c. Set SW1 to 0.

d. Set NOPUT Switch to 1. This switch will be tested whenever a TRAP Table is flushed. If NOPUT is set to 1, SFCHEX will not attempt to input any more commands.

e. Call UNNEED Subroutine to restore the user's program direct bank.

f. Go to A32. This is a location in the PREX Subroutine from which SFCHEX will transfer control to user's program.

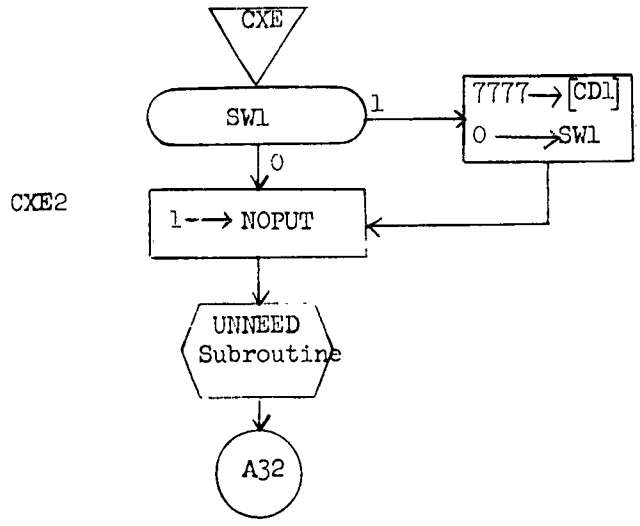

Figure 7.8 CXE

7.2.3.6 CXH Heading Command (Figure 7.9)

This command is to generate a BCD command on the on-line printer or on magnetic tape according to the user's indicated option.

The procedure is:

a. Test SW1. 0 - go to q. 1 - continue.

b. Store the address of the HEAD Subroutine in the TRAP Table via DTR.

c. Increase DTR by one. Store DTR in CD1.

d. Call OVF Subroutine. If the TRAP Table has overflowed, the table will be removed. TT OVRDFLW will appear on the typewriter, and SFCHEX will transfer to BET2 or stop depending on the input mode.

e. Store -40 in CD3.

f. Store BUF+1 in CD2 so that BCD characters can be picked up.

g. Increase CD1 by 1 to leave room in the TRAP Table for the number (divided by two) of BCD characters.

h. Call OVF Subroutine.

i. Pick up BCD character pair via CD2 and store them in the TRAP via CD1.

j. Was one of the characters a period.
Yes - go to step m.
No - continue.

k. Increase CD2 and CD3 by 1.

l. Is CD3 now zero (there are no more characters)
Yes - continue.
No - go back to step g.

m. Store number of character pairs (41 - the number in CD3) in the proper position in the TRAP Table via DTR.

n. Increase CD1 by 1.

o. Call OVF Subroutine.

p. Go to BET2 and input next command.

q. Set DTR to the start of the BCD characters.

r. Call HEAD Subroutine.

s. Go to BET2 and input next command.

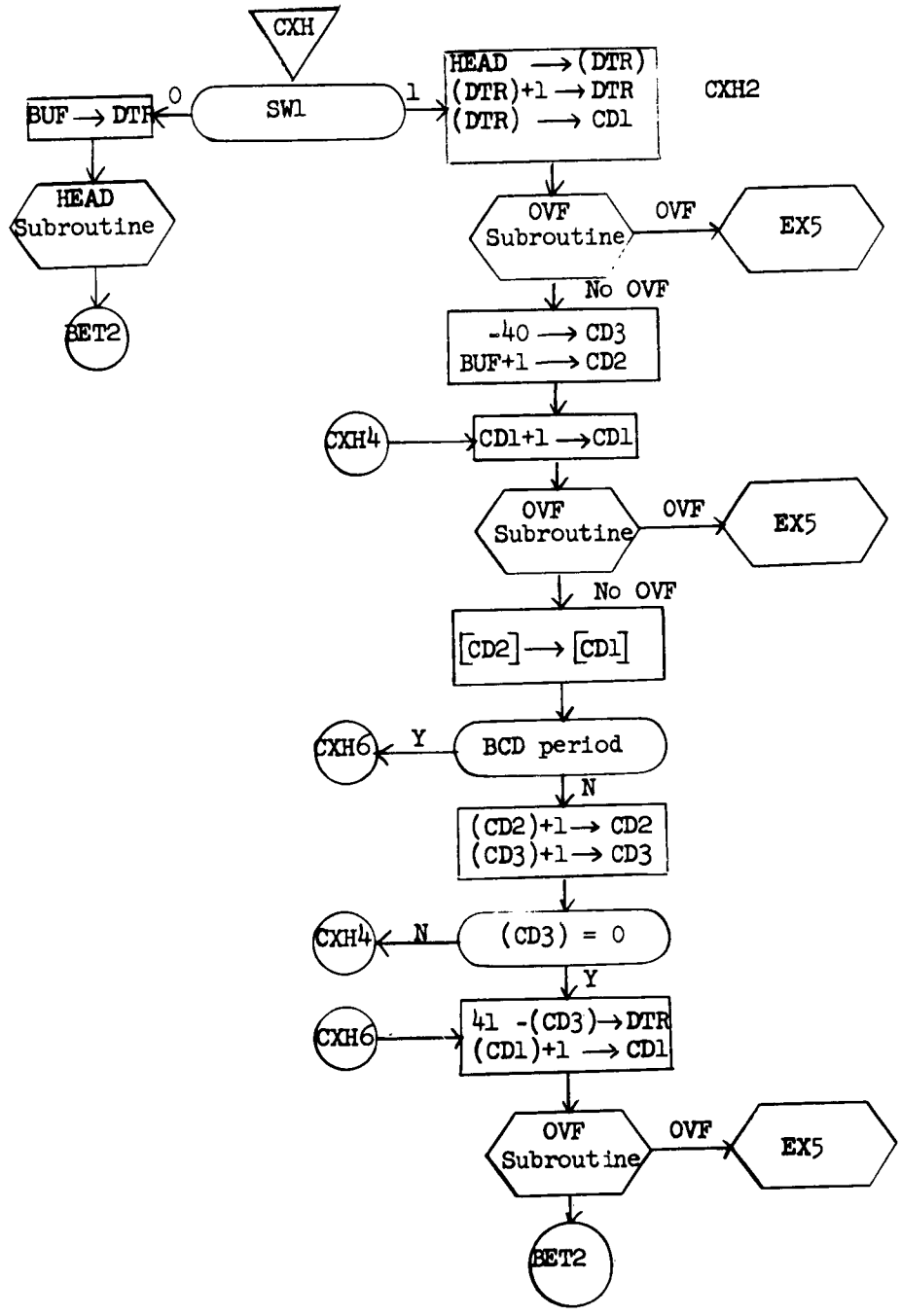

Figure 7.9 CXH

7.2.3.7 CXL LOCATE COMMAND (Figure 7.10)

This command indicates to SFCHEX a location in the user's relative bank in which the RECOV Subroutine (Section 7.3.4.1) may be stored. This command must have been entered before any S, U or T commands.

The procedure is:

a. Call XADD. Now BUF+1 contains the user's relative bank number and BUF+2 the initial address of the RECOV Subroutine.

b. Call LOCATE Subroutine (Section 7.3.4). This subroutine will store the RECOV Subroutine in the user's relative bank and also store certain communication parameters for the intercommunication between SFCHEX and the user's program.

c. Get to BET2 and input the next command.

7.2.3.8 CXP PAGE COMMAND (Figure 7.11)

This command sets up a page eject for the output listing.

The procedure is:

a. Test SW1. 0 - go to step f. 1 - continue.

b. Address of PAGE Subroutine is stored in TRAP Table via DTR.

c. Increase DTR by one.

d. Call OVF Subroutine for possibility of TRAP Table overflow.

e. Go to BET2 and input the next command.

f. Call PAGE Subroutine. PAGE merely causes a page eject (it is not flowcharted).

g. Go to BET2 and input the next command.

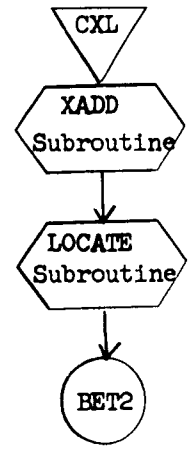

Figure 7.10 CXL

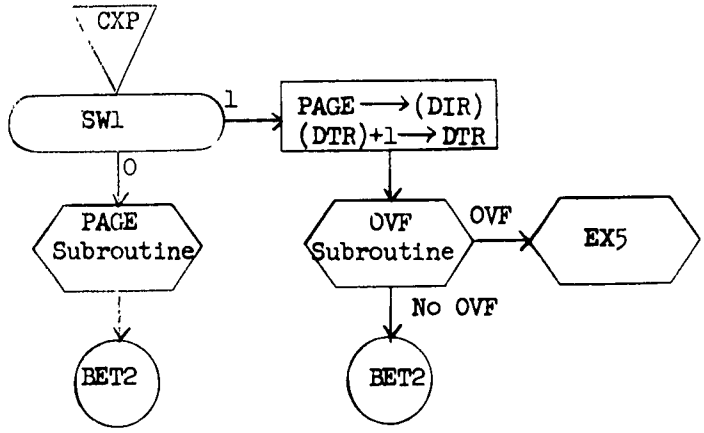

Figure 7.11 CXP

7.2.3.9 CXR RUN COMMAND (Figure 7.12)

This command is used to terminate a sequence of command input.

The procedure is:

a. Test SW1. 0 - go to step d. 1 - continue.

b. Store a 7777 in the TRAP Table via DTR.

c. Set SW1 to 0.

d. Call UNNEED Subroutine to replace user's direct bank.

e. Go to A32 (returns to proper location in user's program).

7.2.3.10 CXS SNAP COMMAND (Figure 7.13)

This command sets up a location in the user's program which, when encountered, causes SFCHEX to dump the direct cells of an indicated bank.

The procedure for setting up this command is:

a. Test SW1. 0 - go to i. 1 - continue.

b. The address of the Snap Setup Subroutine (SNST) is placed in the TRAP Table, via CD1.

c. Increase CD1 by one. Call OVF Subroutine.

d. Call XADD. The bank and location of the snap flag are stored in the TRAP Table via CD1.

e. Increase CD1 by 2. Call OVF Subroutine.

f. Call BIN. This will be the bank to be snapped. It will also be stored in the TRAP Table via CD1.

g. Increase CD1 by 1. Call OVF Subroutine.

h. Go to BET2 and input the next command.

i. Perform steps d through f with the address of the appropriate location in the BUF Buffer initially stored in CD1.

j. Call SNST. This is the SNAP SETUP Subroutine (Section 7.3.5).

k. Go to BET2 and input the next command.

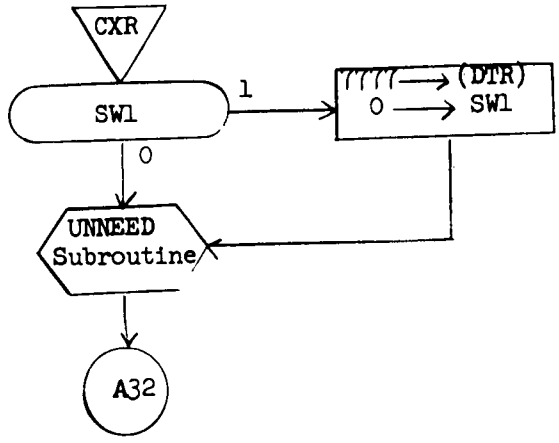

Figure 7.12 CXR

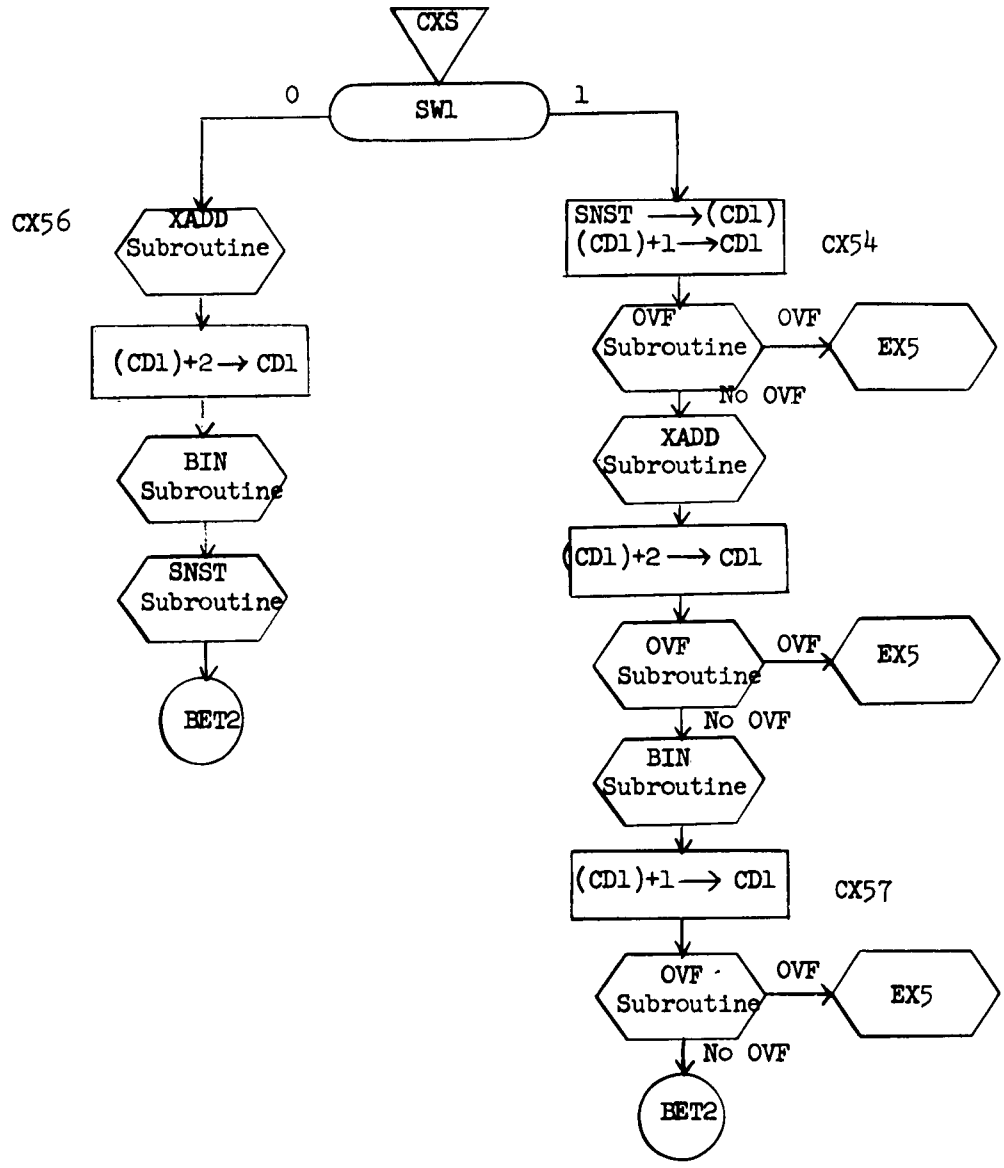

Figure 7.13 CXS

7.2.3.11 CXT TRAP COMMAND (Figure 7.14)

This command indicates that the parameters from subsequent commands are to be stored in a TRAP Table.

The procedure is:

a. Test SW1. 0 - go to step c. 1 - continue.

b. Terminate previous TRAP Table with a 7777 via DTR.

c. Call XADD Subroutine to get Trap Address and bank.

d. Pick up BUF+5. If blank, store -1 in the BUF Buffer. If not blank, use BINC to convert the BCD number to binary.

e. If the BCD number is 0, store a 1 in the BUF Buffer, otherwise store the number.

f. Store a zero in T2+1 and T1 (these are parameters for subsequent subroutines). Set SW1 to 1.

g. Call INSV (Section 7.3.6). This subroutine saves the instructions which are removed from the user's program, stores the jump to RECOV in that location, picks an available TRAP Table and initializes DTR.

h. Call SETUP (Section 7.3.7). This subroutine translates the saved instructions so that they can be executed in the bank in which SFCHEX is located.

i. Go to RET2 and input the next command.

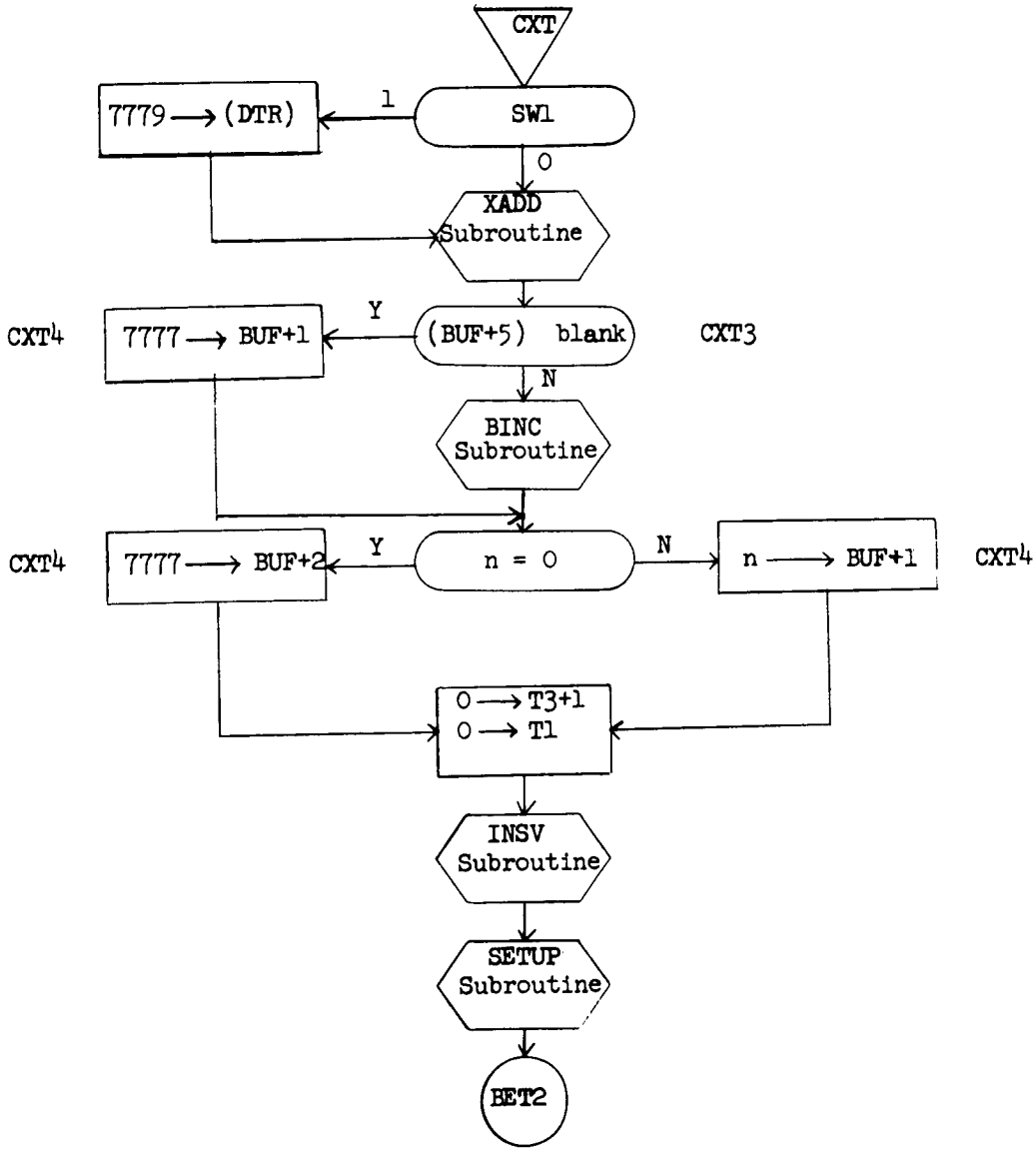

Figure 7.14 CXT

7.2.3.12 CXU UNSNAP Command (Figure 7.15)

This command sets up a location in the user's program which, when encountered, will cause a snap location in the user's program to be removed.

The procedure is:

a. Test SW1. 0 - go to step i. 1 - continue.

b. Address of UNST Subroutine is stored in the TRAP Table via CD1.

c. Increase CD1 by one. Call OVF Subroutine.

d. Call XADD - the unsnap address and bank will be placed in the TRAP Table.

e. Increase CD1 by two. Call OVF Subroutine.

f. Call XADDA - the address and bank of the snap address which is to be removed will be placed in the TRAP Table.

g. Increase CD1 by one. Call OVF Subroutine.

h. Go to BET2 and input the next command.

i. Call XADD - unsnap bank is in BUF+1, unsnap address is in BUF+2.

j. Call XADDA - bank of snap command to be removed is in BUF+3. Address is in BUF+4.

k. Call UNST (Section 7.3.8). This subroutine sets up a flag jump in the user's program for the unsnap execution.

l. Go to BET2 and input next command.

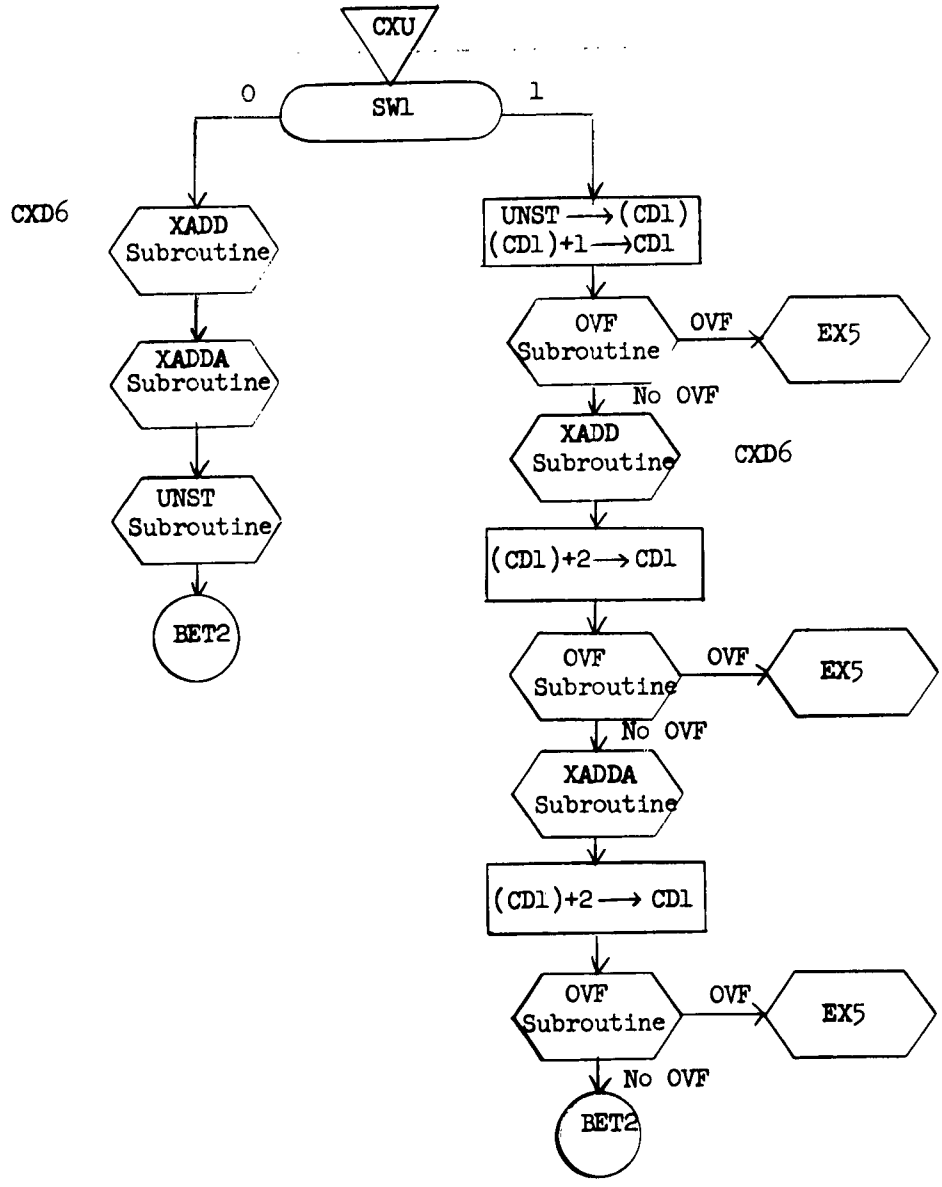

Figure 7.15 CXU

7.2.3.13 CXW WRAPUP Command (Figure 7.16)

This command causes the termination of input as well as the termination of the program when all the (finite) TRAP TABLES have been flushed.

The procedure is:

a. Test SW1. 0 - go to step d. 1 - continue.

b. Store a 7777 in the TRAP Table via CD1.

c. Store a 0 in SW1.

d. Set NOPUT Switch to 1. Set TERM Switch to 1. The TERM switch initiates the test of the TRAP Tables.

e. Call UNNEED Subroutine to return user's program direct cells.

f. Pick up TT. If the contents of TT do not equal 0 or -1, go to A32. If they do equal 0 or -1, continue.

g. If TT+50 contains 0 or -1, stop. Otherwise, go to A32.

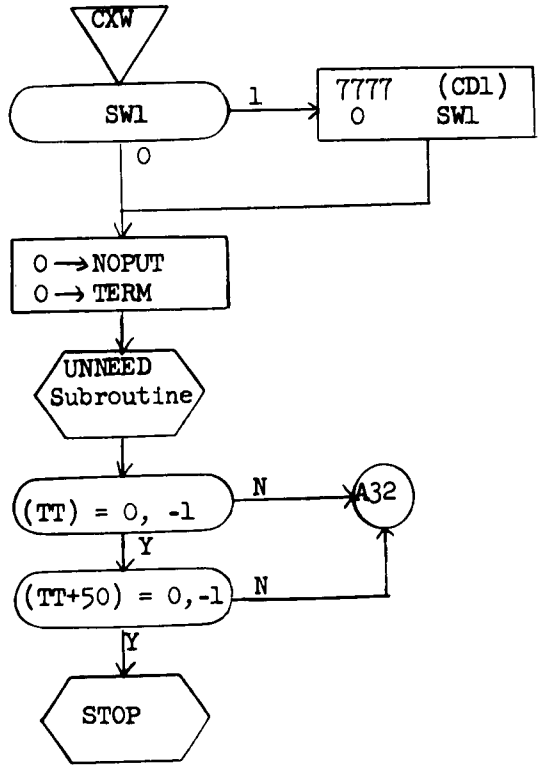

Figure 7.16 CXW

7.2.3.14 CXN Mnemonic dump (Figure 7.17)

This command causes core to be printed in 60 word blocks without conversion.

The procedure is:

a. Change the references to the DUMP subroutine to ALPHAD, which is the address of mnemonic dump subroutine. The changes are made in CXD routine.

b. Transfer to CXD.

c. CXD restores the DUMP address after executing the mnemonic dump.

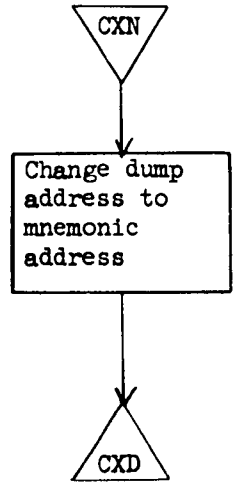

Figure 7.17 CXN

7.2.4 OVF and EX5

These two subroutines appearing in many flow charts are relatively simple and do not require a flow chart.

OVF Subroutine compares the address in CD1 with the address of the limit of the Trap execution table. If the limit is exceeded, the Trap is wiped out by replacing the instructions in the user's program (RECOV Subroutine), and placing a zero in SW1.

The EX5 is called to print out

TT OVRFLW

If the input mode is via typewriter, an asterisk will be printed and the user may reenter a modified table. If the input mode is via punched card, the program will stop to allow the user to modify the card sequence.

7.3 Common Subroutines

The following set of subroutines are used in both the setup and execution of the instructions as they are read in (Section 7.2) and the execution of the TRAP Tables.

7.3.1 CORR Correct Subroutine (Figure 7.17A)

This subroutine places the corrections in the proper location. The storage sequence in either the BUF Buffer or a TRAP Table is

A	1	bank
A	2	address
A	3	N - number of corrections
A	4	corrections
:	:	
A	+N+3	correction

The procedure is:

a. Decrease DTR by 1.

b. Compute -(N+3) and store the result in the TRAP Table via DTR. This will cause subsequent TRAP executions to bypass this region.

c. Increase DTR by 1, and pick up the bank number (A1). Store the set bank instruction in CR1+2.

d. Increase DTR by 1 and pick up the address (A2). Store it in CR1+4.

e. Increase DTR by 1 and store A3 (the number of corrections) in CR2.

f. Increase DTR by 1 and store the contents of DTR in CR1+1.

g. Add CR2 to DTR (for the execution of the next portion of the TRAP Table.

h. Pick up the corrections via CR1+1.

i. Set to user's bank in CR1+2.

j. Store the corrections via CR1+4.

k. Increase the address in CR1+1 and CR1+4.

l. Decrease CR2 by 1. Test CR2. 0 - exit. Nonzero - return to step h.

7.3.2 DUMP Subroutine (Figure 7.18)

This subroutine can be called directly from the input section, from a TRAP Table of from the SNAP Subroutine (Section 7.4.6). The sequence of setup parameters is the same in each case.

A1	bank 1	
A2	address 1	start of dump
A3	bank 2	
A4	address 2	end of dump

The procedure is:

a. The address A1 is contained in DTR. Bank 1 is stored in DBNKS via D1.

b. Increase DTR and D1 by 1. Store address 1 in DST via D1.

c. Increase DTR and D1 by 1. Store bank 2 in DBNKT via D1.

d. Increase DTR and D1 by 1. Store address in DTERM via D1.

e. If DSPFLG is zero, the DUMP Subroutine was called from a TRAP Table or from a SNAP Subroutine. In this case the B D I R A P BER heading is generated in the bank setting. A-Register, instructions address and Buffer Entrance Register values are printed.

f. UNNEED Subroutine is called in order to set the right parameters in the direct cells.

g. Subtract DBNKS from DBNKT. If non-zero go to step h, otherwise store 1 in FLG77, store DTERM in D38 and go to step i.

h. Store 0 in FLG77 and 7777 in D38.

i. Truncate the address in DST to the nearest multiple of 16. (This is done to give even octal 20 address down the left side of the dump.)

j. Pick up and convert to BCD, 16 cells starting with the address in DST. Increase DST by 20.

k. Output the line, and

l. Subtract DST from D39. Positive go to step 10 - zero or negative continue.

m. Test FLG77 0 - Call NEED Subroutine and exit.
1 - Store 0 in DST, increase DBNKS by 1 and go to step g.

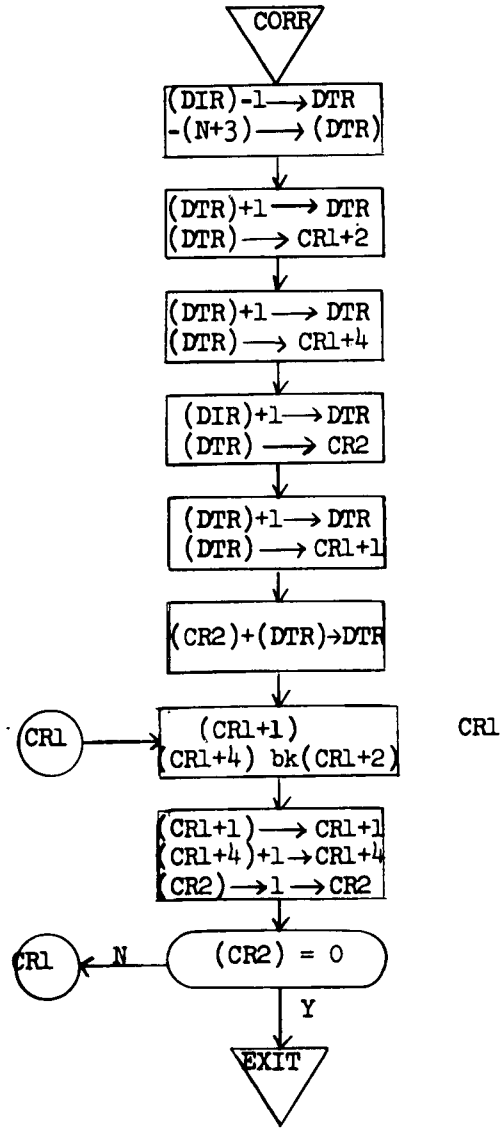

Figure 7.17A CORR

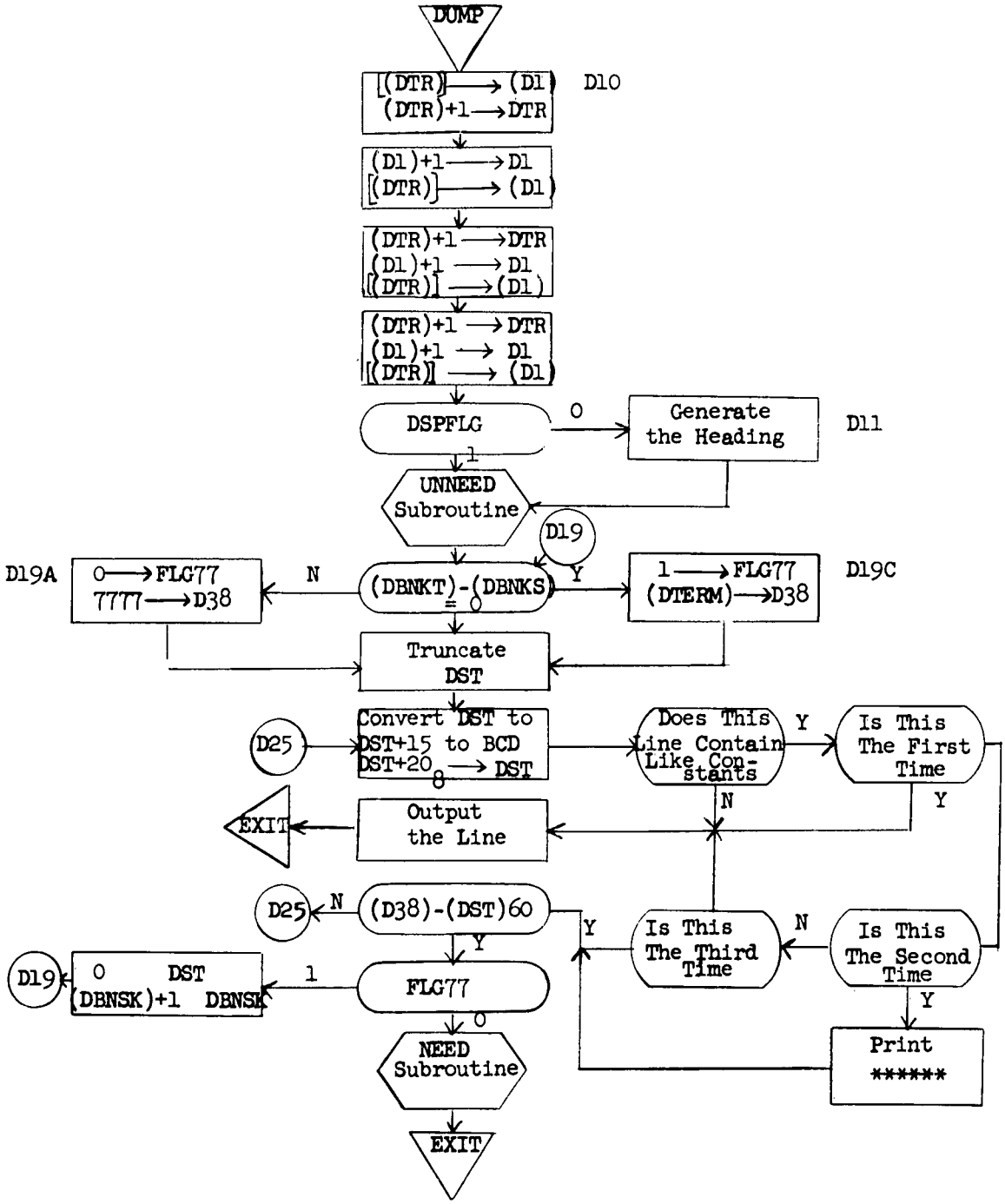

Figure 7.18 DUMP

7.3.3 HEAD Subroutine (Figure 7.19)

This subroutine outputs the BCD annotations which were input via the H Command.

The sequence of parameters is the same for the two sections which call the HEAD Subroutine.

BUF Buffer		TRAP Table	
A1	39	A1	N
A2	H_1 H_2	A2	H_1 H_2
.		.	
.		.	
.		.	
A_{N+1}	H_{2N-1} H_{2N}	A_{N+1}	H_{2N-1} H_{2N}

In the BUF Buffer, the whole buffer is output with the following blanks.

The procedure is:

a. DTR contains the address A1. Store the contents of A1 in CD1 via DTR. Store the initial address of the Output Buffer (OUTBUF) in CD2.

b. Store the contents of A1 in CD3 via CD1. Increase CD1 by 1.

c. Transfer characters to OUTBUF using CD1 and CD2.

d. Increase CD1, CD2 and CD3 by 1.

e. Test CD3 0 - go to step f
1 - go to step c

f. Output the annotation.

g. Exit.

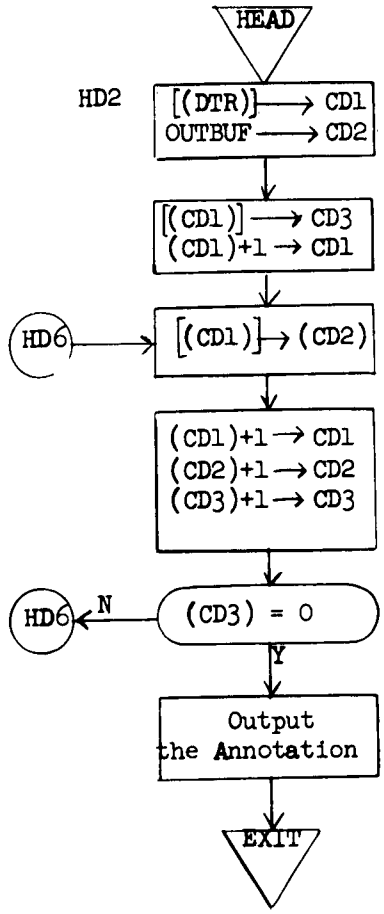

Figure 7.19 HEAD

7.3.4 LOC Locate Subroutine (Figure 7.20)

This subroutine transfers the RECOV Subroutine (Section 7.3.4.1) to the address and bank indicated by the L command. (Note: This subroutine can only be called from the Input Section.)

The calling sequence will be in the BUF Buffer.

BUF+1	Bank
BUF+2	Address

The procedure is:

a. Store the bank in BUF+1 in UR and CL2+1. Store the address in BUF+2 in various return addresses.

b. Store 9 in LC3.

c. The first cell of RECOV is RCS. Pick up the contents of RCX and store it in the user's program via LC2+1.

d. Decrease LC3 by 1 and increase RCX by 1.

e. Test LC3. 0 - return to BET2.
Nonzero - to step c.

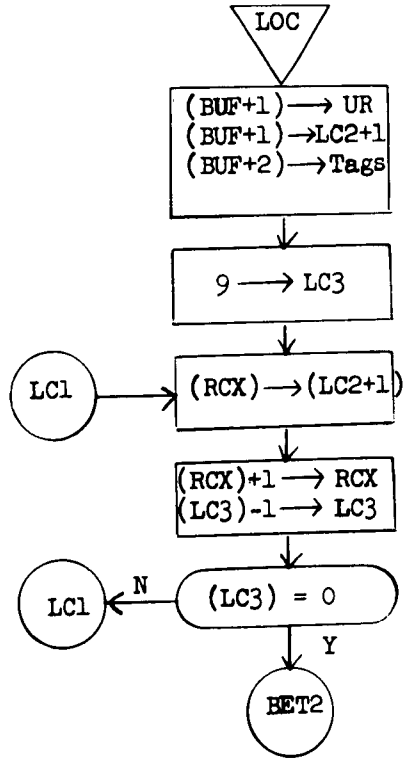

Figure 7.20 LOC

7.3.4.1 RECOV Subroutine (Figure 7.21)

The RECOV Subroutine is the link between the user's program and SFCHEX.

AS, T or U command causes a JPR to RECOV to be stored in 2 cells at an indicated address. With that address the following procedure is followed:

a. Save A-Register in RECOVA.

b. Go to SFCHEX ENTRY (executes S, T or U command).

c. Restore A-Register (may have been modified by SFCHEX).

d. Clear lockout.

e. Return to user's program at location set by SFCHEX.

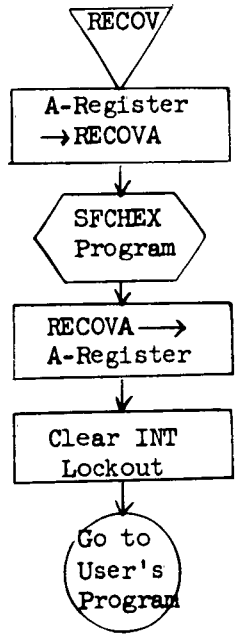

Figure 7.21 RECOV

7.3.5 SNST Subroutine (Figure 7.22)

This subroutine sets up the SNAP command in the user's program.

The calling sequence is either in the TRAP Table or in the BUF Buffer.

A1 bank
A2 address
A3 bank to be snapped

Procedure:

a. Set T3+1 to 12 and T1 to 2. These are parameter settings for the following subroutines.

b. -4 is stored via DTR. If the calling sequence is in the TRAP Table, the -4 will cause the TRAP execution to bypass this section of the TRAP Table, next time through. If the sequence is in the BUF Buffer, no harm is done.

c. Call INSV. INSV will save the instructions in the user's program, set the jump to RECOV in the user's program, store the bank to be snapped and return.

d. Call SETUP. This subroutine will translate the instructions which were saved, so that they may be executed in SFCHEX' relative bank.

e. Return.

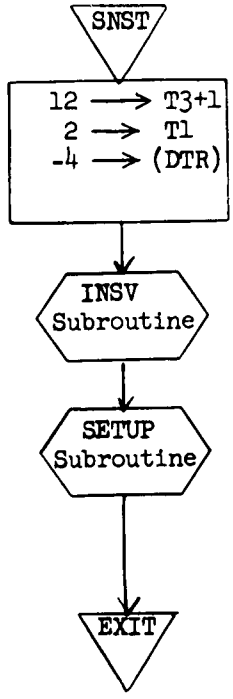

Figure 7.22 SNST

7.3.6 INSV Subroutine (Figure 7.23)

This subroutine saves the two cell locations in the user's program, which are to be replaced by a JPR to RECOV.

The following two parameters are set.

T1 = 0 Saving for a T Command
= 1 Saving for a U Command
= 2 Saving for a S Command

T3+1 = 0 for a T Command
= 6 for a U Command
= 12 for an S Command

Tab Buffer

The Tab Buffer consists of 18 cells divided into 3 six cell sections. These six cell sections are in turn divided into two 3 cell sections. Each of these 3 cell sections may contain an address and the contents of two cells from the user's program. The six cell sections can contain up to two sets of saved instructions. Thus T3+2 is set so that INSV will immediately check the proper portion of TAB for available locations. (0-5 for T, 6-11 for U, 12-17 for S).

The bank and address at which the jump to RECOV is to be placed are either in the BUF Buffer or a TRAP Table (in the case of a T command, they could only be in the BUF Buffer).

	TRAP	UNSNAP	SNAP
A1	bank	bank 1	bankk
A2	address	address 1	address
A3	n	bank 2	snap bank
A4		address 2	

Procedure:

a. Get initial position in the TAB buffer by adding the initial address of TAB to the contents of T3+1 and store the result in T3+1.

b. Increase DTR by 1. DTR now contains the address A2.

c. Pick up the contents of A2 and store it in CD2, increase CD2 by 1 and store it in CD3.

d. Store a -1 in CD4.

e. Pick up the location in the TAB Buffer identified by T3+1. If 7777,go to step g. If not 7777,continue.

f. Increase CD4 by 1. If 0, increase T3+1 by 3 and return to step e. If 1, then we already have the TAB Buffer filled with this type of command. EX5 is called to print out THIRD COM.

g. Store this available TAB address in CD5. Increase it by 1 and store in CD6. Increase CD6 by 1 and store in CD7.

i. Pick up the instructions in the user's program using CD2 and CD3. The instructions are stored in IN2 and IN2+1.

j. Store the JPR instructions in the user's program via CD2 and the RECOV Subroutine address via CD3.

k. Store the address in CD2 in TAB via CD5. Store the instruction in IN2 in TAB via CD6. Store the instruction in IN2+1 in TAB via CD7.

l. Test T1. If 0 (TRAP) go to step 17.
If 1 (SNAP) go to step 14.
If 2 (UNSNAP) continue.

m. Increase DTR by 2. Pick up the address of the UNSTBL+1, add it to CD4 and store the bank (whose address is in DTR) in the UNSTBL Table via CD4.

n. Exit.

o. Increase DTR by 1. Pick up the address of the SNTB+1, add it to CD4 and store the result in CD5. Pick up the snap bank (whose address is in DTR) and store it in SNTB via CD5.

p. Store the address if the SX Table in A2. Test CD4. If negative increase A2 by 30 and then go on.

r. Exit.

s. Store the address of the TX Table in A2. Store the address of the TT Table in DTR. Test CD4. If negative increase A2 by 30, DTR by 50. Pick up n from BUF+3 and store it in the TRAP Table via DTR.

t. Exit.

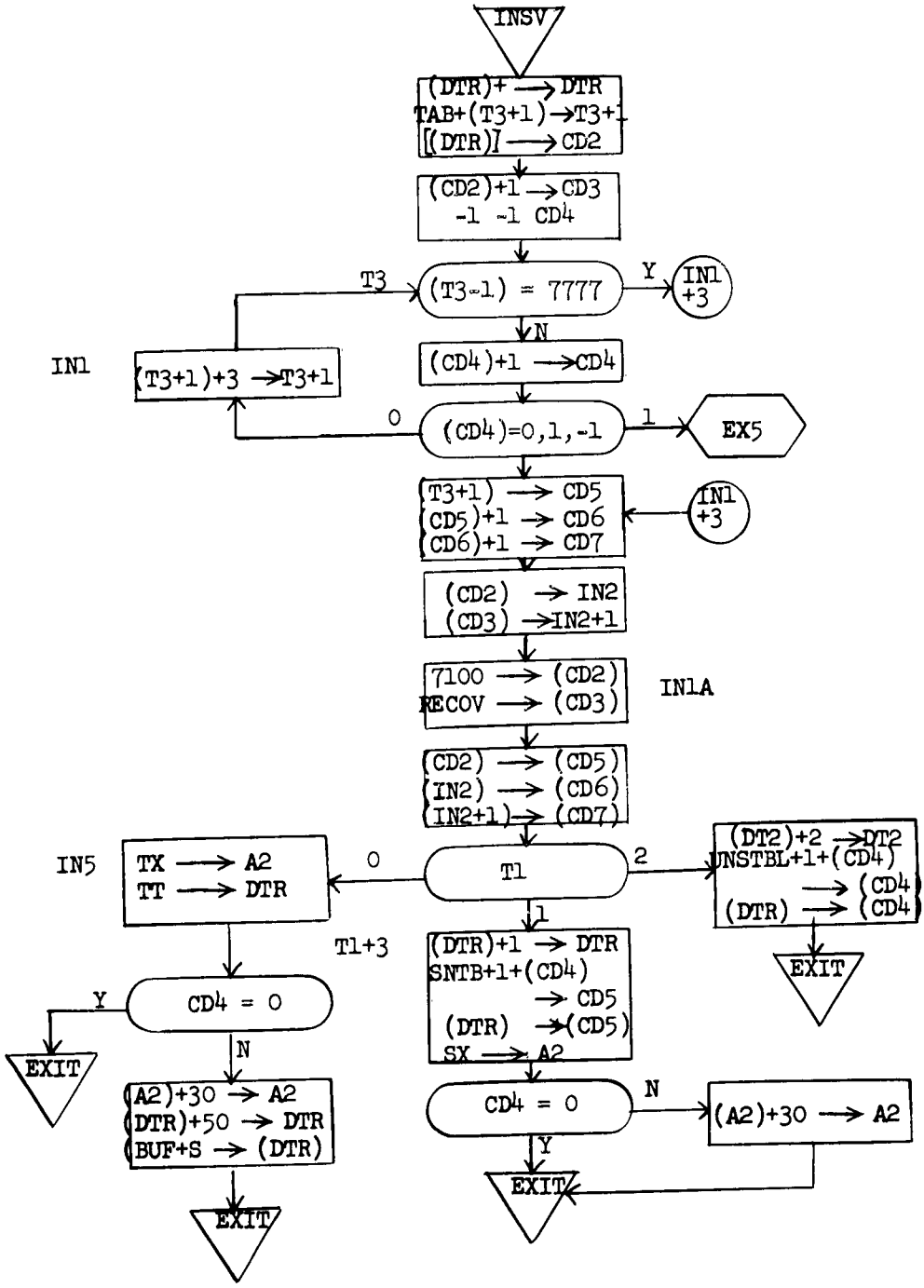

Figure 7.23 INSV

7.3.7 SETUP Subroutine (Figures 7.24 through 7.39)

This subroutine is used in setting up a SNAP or TRAP jump in the user's program. Since a SNAP or TRAP jump may remain in the user's program through many iterations, it is necessary to translate the saved commands into instructions whose relative bank is the same as SFCHEX.

INSV sets up the two parameters which SETUP will need.

a. T3+1 the location in TAB of the two saved cells with their corresponding address.

b. A2. The available location for the translation in either the TX or SX Buffer.

SETUP divides into nine modules, each of which translates a certain class of instructions. These classes are defined with regard to basic similarities in their octal code representations.

For each module, a translation sequence of instructions is already coded. However, SFCHEX must complete those instructions with the proper addresses, bank numbers, etc. When the sequence is established, the subroutine GEN is used to transfer the entire sequence to A2 through A2+n, when n+1 is the number of instructions.

7.3.7.1 GEN Subroutine (Figure 7.24)

As stated above, this subroutine transfers a sequence of instructions from a module storage buffer to the TX or SX Buffer using the address in CD4.

GEN uses the following calling sequence:

A1 JPR
A2 GEN
A3 n (number of instructions)
A4 initial address of instruction sequence in module buffer.

Procedure:

a. Transfer the contents of the address contained in A4 to GEN3+1.

b. Transfer the number in A3 to GENN.

c. Transfer the instruction to the execution table via CD4.

d. Increase CD4 by one. (GEN3+1) by one.

e. Decrease GENN by 1.

f. Test GENN 0 - return
1 return to step c.

7.3.7.2 WT Subroutine (Figure 7.25)

In the translation of 2 cell instructions, it is sometimes necessary to go down into the user's program and pick up the second cell. This is done through the WT Subroutine. Its calling sequence is:

```
1    JPR
2    WT
```

The A-Register must contain the address in the module translation buffer in which the cell is to be stored.

Procedure:

a. Store the address in the A-Register in WT1+3.

b. Test CD6. 0 means that we are currently processing the first cell, so that the second cell is already saved in the TAB Buffer. Store the address contained in SEA+1 in WT1+1.
Go to step e.

1 means that we are processing the second cell in the TAB Buffer, and the next cell must be picked up from the user's program.
Continue to step c.

c. Increase EXIT by 1 via CD9, so that return to SFCHEX will occur one cell past the cell we are picking up.

d. Pick up cell in user's program via CD5. Go to step g.

e. Set CD6 to 1 (this will indicate that the second cell has been processed.)

f. Pick up the second cell which was stored in the TAB Buffer via WT1+1.

g. Store the cell at the address contained in WT1+3.

h. Return.

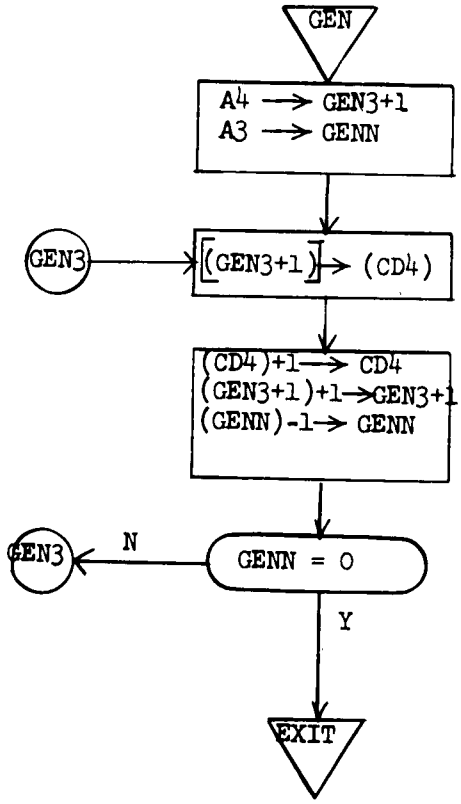

Figure 7.24 GEN

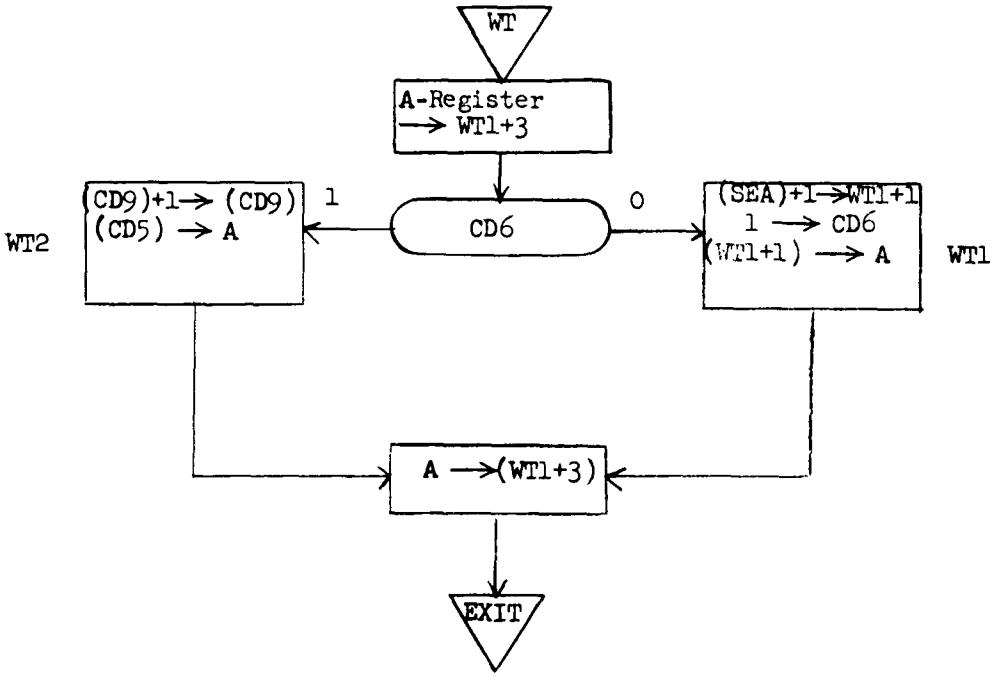

Figure 7.25 WT

7.3.7.3 SETUP Module Control Sequence (Figure 7.26)

This section analyzes the cell to be translated and transfers control to the proper module. SETUP will be provided with the locations of the cells to be translated and the place in the TX or SX Buffer where the translation is to be stored (A2).

a. Store 0 in CD9 (Translating first cell). Store the contents of A1 (address of first instruction in TAB) in SEA+1, Pick up the first instruction via SEA+1. Store in CD1.

b. Divide the cell into the E anf F fields. E goes to CD2 and F to CD3.

c. Test CD 3 0 - go to step j.
non 0- continue.

d. Test CD3 If CD3 is greater than 70 go to step m.
If CD3 is less than or equal to 70 continue.

e. Test CD3 If CD3 is less than 60 go to step f.
If CD3 is greater than or equal to 60 we have a conditional jump instruction since CD3 is greater than 60 and less than 70 or a JPI instruction if CD3 equals 70. Both these types of commands are handled in Module 2. (Section 7.3.7.3.2).

f. Test the complete cell (CD1) for 0101. This would be a PTA instruction. If the test is negative, go to step g, otherwise go to Module 9 (Section 7.3.7.3.9) for the PTA translation.

g. Test the complete cell (CD1) for 0100 (BLS). 0105 (ATE) or 0106 (ATX). If the tests are negative, go to step h, otherwise go to Module 7 (Section 7.3.7.3.7).

h. Test the complete cell (CD1) for 0104 (CBC) or 0130 (CIL). If the tests are negative, go to step i, otherwise go to EX1. These are illegal commands and EX1 will point on the typewriter "IL INST AT bXXXX", then SFCHEX halts.

i. Test the complete cell (CD1) for the format 015X (STP) instruction. If the test is negative, the cell contains a command which is translated by Module 1 (Section 7.3.7.3.1). If the test is positive, the STP instruction is translated by Module 8 (Section 7.3.7.3.8).

j. (F = 0 in step c) Test CD2
0 - means we have a HLT Command. This is illegal. EX1 is called and SFCHEX halts.
Non-0, continue.

k. Test CD2. If CD2 is not greater than seven, go to Module 1. (This is a NOP). If CD2 is not greater than seven - continue.

l. Shift the contents of CD2 eight places to the left. If the result is negative, the cell contains a bank set and jump. This instruction is translated by Module 5. (Section 7.3.7.3.5). If the result is positive, the cell contains a bank instruction. This will be translated by Module 6. (Section 7.3.7.3.6).

m. (CD3 greater than 70 in step d).
Test CD3. If CD3 equals 71 and CD2 equals 0, we have an illegal instruction (JPR). EX1 will print its comment and SFCHEX will halt. If CD3 equals 71 and CD2 doesn't equal 0, we have a JFI command which is translated by Module 3 (Section 7.3.7.3.3). If CD3 doesn't equal 71, continue to step n.

n. Finally, test the cell (CD1) for a 77X0 (SLJ). If the test is negative, we have an illegal instruction, EX1 prints its comment and SFCHEX halts.

   If the test is positive, Module 4 (Section 7.3.7.3.4) translates the SLJ Command.

o. Each time a Module completes a translation, it returns to ALPHA (upper left portion in Figure 7.26). ALPHA is step p.

p. Test CD6. 0 - means another cell to translate. Set CD6 to 1, return to step 2.
   1 - means the second cell has been translated. At this time a final sequence of commands which make sure that when the translation is executed the final instruction will transfer to A31 in PREX, is stored via CD4.

q. Return.

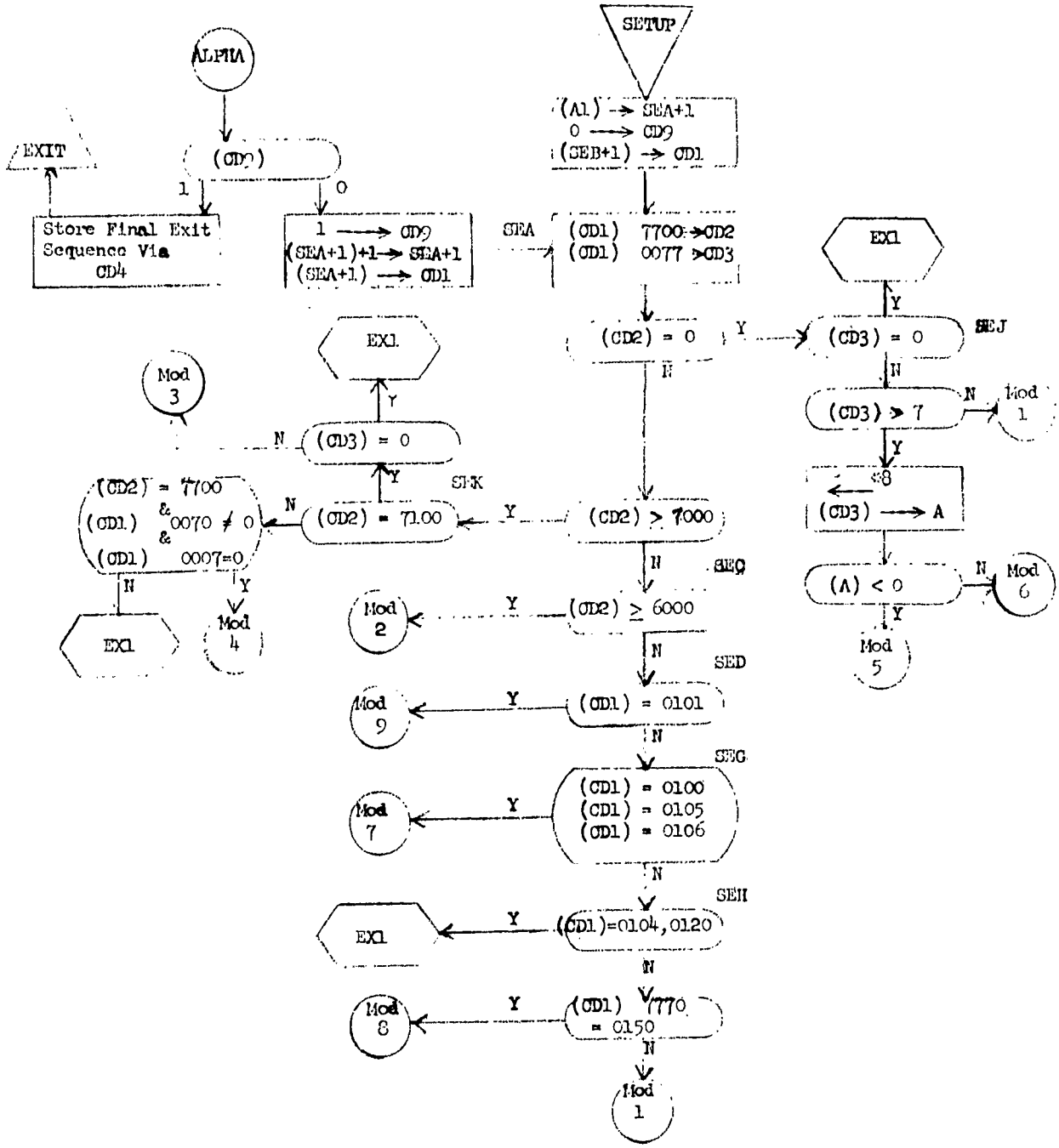

Figure 7.26 SETUP

7.3.7.3.1 MODULE 1 (Figure 7.27)

This module translates Relative Instructions, Constant Instructions, Two Word Instructions other than constant and any other instructions not handled by the other Modules.

Procedure:

a. Set the translation exit so that the result in the A-register will be saved. This is done by storing A31 through CD12.

b. Pick up the units digit of CD2, store the result in CD13.

c. Test the instruction for form 0XXX. If the test is negative, go to step d. If the test is positive, go to Module 1.1.

d. Test CD3. If CD3 does not equal 0, we have a relative mode, direct or indirect instruction, treated in Module 1.2. If CD3 equals 0 - continue.

e. If CD13 equals 3 or 7, go to Module 1.1 (this indicates a specific command since E equaled 0).

f. If CD13 equals 2 or 6, go to Module 1.4 (this indicates a constant command).

g. Otherwise we have a two word instruction which is handled by Module 1.3.

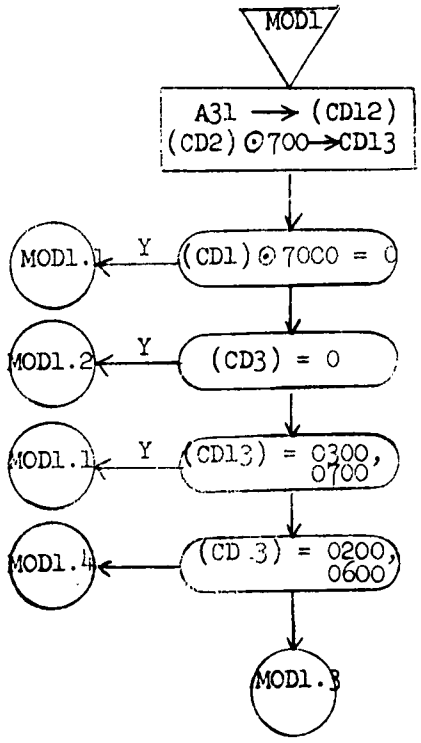

Figure 7.27 MODULE 1

7.3.7.3.1.1 MODULE 1.1 (Figure 7.28)

This section translates all the single cell arithmetic, direct and specific instructions. The translation sequence for any such command is:

T13	SIC	SFCHEX Relative
	+1	LDM
	+2	BN3 (BN3 is a SIC to the user's current
	+3	STF3 indirect setting)
	+4	LDM
	+5	SA (user's A-Register)
	+6	0 (stored from above)
	+7	INSTRUCTION
	+8	SIC SFCHEX Relative
	+9	STM
	+10	SA (new user's A-Register)

The procedure is:

a. Store CD1 in T13+7.

b. Call GEN with a calling sequence of 11, T13.

c. Return to ALPHA (Figure 7.26).

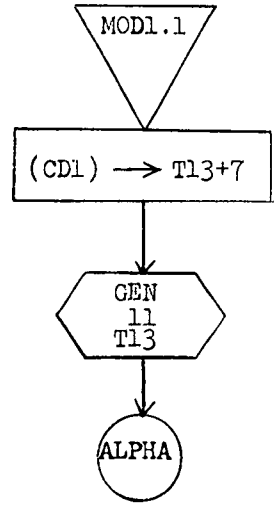

Figure 7.28 MODULE 1.1

7.3.7.3.1.2 MODULE 1.2 (Figure 7.29)

This section translates all relative mode instructions. The translation sequence for any such command is:

T11	SIC UR	(User's Relative)
+1	XXM	(Change the relative command to a memory command)
+2	(p) $\pm$ E	
+3	SIC	SFCHEX Relative
+4	STM	
+5	SA	

Procedure:

a. If CD13 = 200, 600 Subtract 100 from SC1 and store it in T11+2. This changes the relative forward instructions to a memory instruction. Go to step d.

b. If CD13 = 300, 700 Subtract 200 from CD1 and store in T11+1. This changes the relative backward instruction to a memory instruction. Go to step e.

c. If CD13 was neither of the above, go to Module 1.1.

d. Add CD5 (the address of the instruction) and CD3 (the E field) Store in T11+2. Go to step h.

e. Subtract CD3 from CD5, store in T11+2.

f. Store the SIC UR in T11.

g. Call GEN with sequence 6, T11.

h. Go to ALPHA.

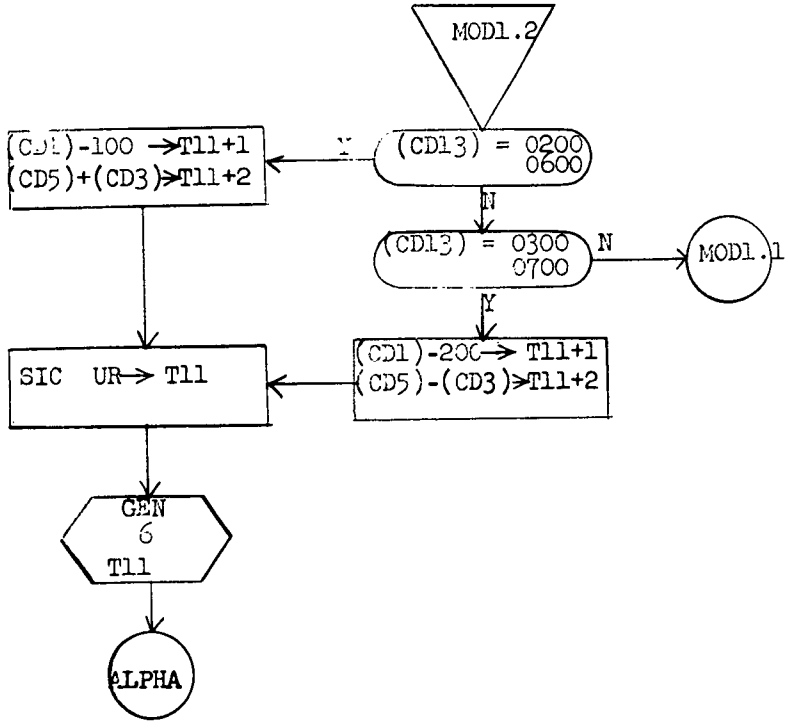

Figure 7.29 MODULE 1.2

7.3.7.3.1.3 MODULE 1.3 (Figure 7.30)

This section translates constant mode instructions. The translation sequence for such a command is:

T11	INSTRUCTION
+1	2nd CDLL obtained by WT Subroutine
+2	SIC SFCHEX Relative
+3	STM
+4	SA Save A-Register

Procedure:

a. Store CD1 in T11.

b. CALL WT with address T11+1 in the A-Register.

c. CALL GEN with calling sequence 5, T11.

d. Go to ALPHA.

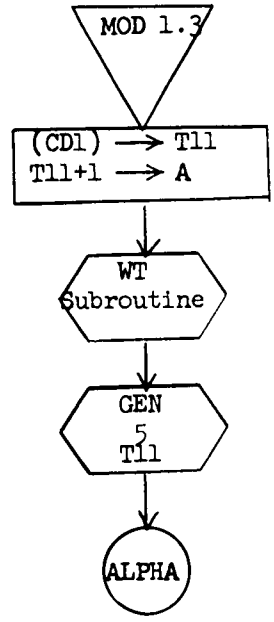

Figure 7.30 MODULE 1.3

7.3.7.3.1.4 MODULE 1.4 (Figure 7.31)

This section translates the memory instructions. The translation sequence is:

T14	INSTRUCTION
+1	Second cell using WT.
+2	SIC SFCHEX Relative.
+4	STM
+5	SA Save A-Register.

The procedure is:

a. Store CD1 in T14.

b. Call WT Subroutine with T14+1 in the A-Register.

c. Call GEN with calling sequence 5, T14.

d. Go to ALPHA.

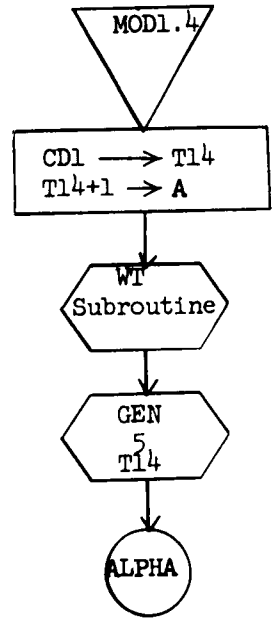

Figure 7.31 MODULE 1.4

7.3.7.3.2 MODULE 2 (Figure 7.32)

This module translates conditional jumps and the JPI command. The translation sequences are:

CON. JUMP

T21	SIC	SFCHEX	BANK
+1	XXF	2	Same condition as INST
+2	YYF	7	Opposite condition
+3	LDC		
+4		(P) ± (E)	Jump address
+5	STM		SFCHEX will return to
+6		EXIT	the address in EXIT
+7	JF1	1	
+8		A32	Current A-Register won't be saved.

JPI INSTRUCTION

T22	SIC	SFCHEX	BANK
+1	LDD(E)		
+2	STM		SFCHEX will return to the
+3	EXIT		address in EXIT.
+4	JFI	1	Current A-register won't
+5		A32	be saved.

Procedure:

a. Test CD2. CD2 less than 6400 means a conditional forward jump Store CD5 + CD3 in T21+4. Go to step e.

b. Test CD2. CD2 = 7000 means a JPI instruction - go to step i.

c. If the above tests fail, we have a conditional backward jump. Store CD5 - CD2 in T21+4.

d. Subtract 400 from CD2. This makes it a relative forward jump. Store the result in CD2.

e. Add 2 and store in T21+1.

f. Shift CD2 by 5. Test result.
Negative - Add 0107 to CD2 and store in T21+2.
Positive - Subtract 0071 from CD2 and store in T21+2.
The result is the opposite conditional jump.

g. Call GEN with calling sequence 9. T21.

h. Go to ALPHA.

i. Add 2000 to CD3, this will be a LDD Command. Store the result in T22+1.

j. Call GEN with calling sequence 6, T22.

k. Set CD6 to 1, since it will be unnecessary to translate a second instruction.

l. Go to ALPHA.

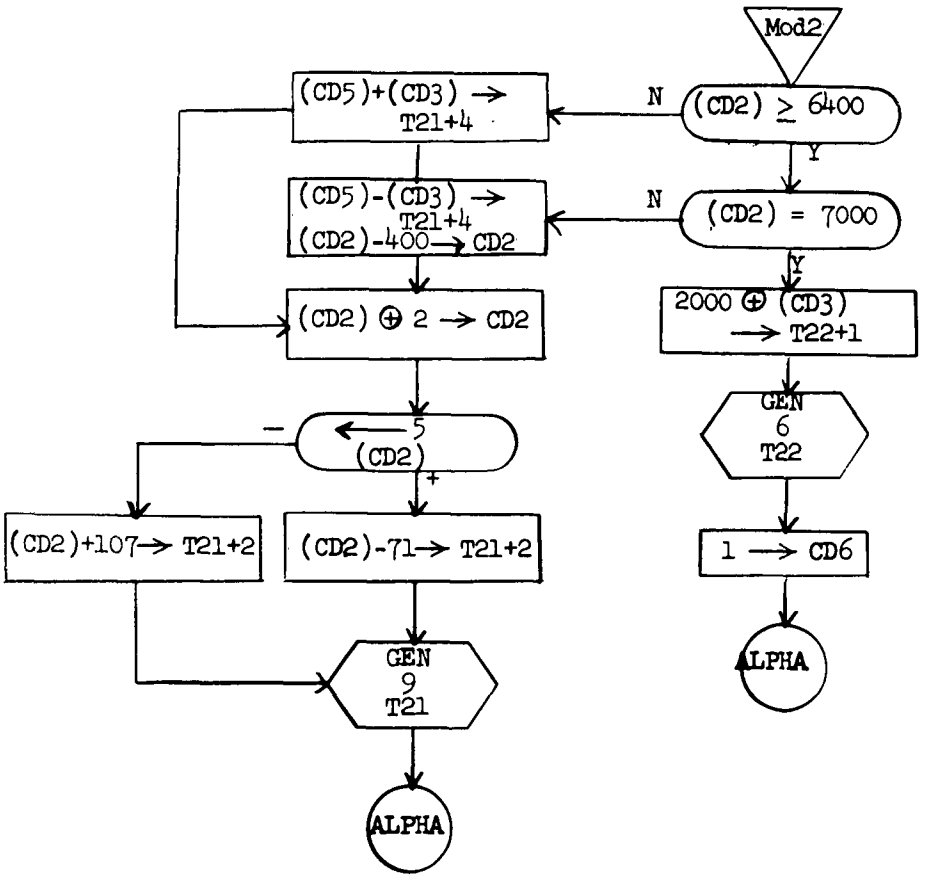

Figure 7.32 MODULE 2

7.3.7.3.3 MODULE 3 (Figure 7.33)

This module translates the JFI instruction. The translation sequence is:

T31	LDC
+1	(p) + (E)
+2	SIC SFCHEX BANK
+3	STM
+4	EXIT
+5	JFI 1
+6	A32

Procedure:

a. Add (CD6) to (CD3) and store in T31+1.

b. Call GEN with calling sequence 7, T31.

c. Set CD6 to 1.

d. Go to ALPHA.

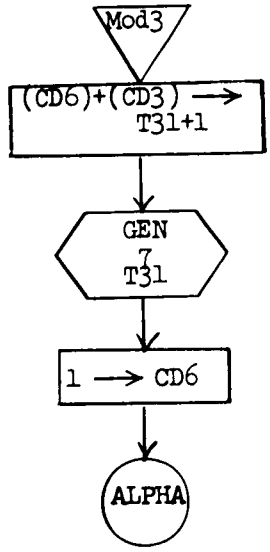

Figure 7.33 MODULE 3

7.3.7.3.4 MODULE 4 (Figure 7.34)

This module translates the SLJ Instruction. The translation sequence is:

T_4	INSTRUCTION		
+1	*+3		Just jump to T_4+4 if condition holds.
+2	NZF	9	These two instructions get out of this
+3	ZJF	8	sequence.
+4	SIC	SFCHEX BANK	
+5	LDC	0	
+6		G	Obtained by WT
+7	STM		
+8	EXIT		
+9	JFI	1	
+10	A32		

Procedure:

a. Store CD1 in T_4. Store the address in CD4 increased by 4. T_4+1 for the jump to T_4+4 in the execution table.

b. Call WT to pick up second cell (T_4+6 in A-Register)

c. Call GEN with calling sequence 11. T_4.

d. Return to ALPHA.

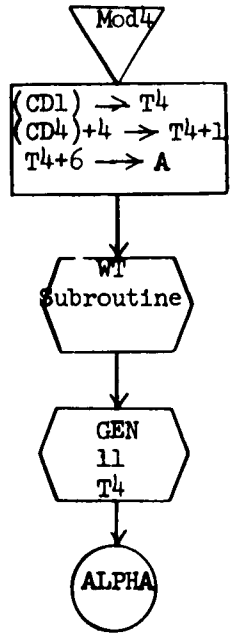

Figure 7.34 MODULE 4

7.3.7.3.5 MODULE 5 (Figure 7.35)

This module translates bank sets and jumps. The translation sequence is:

T5	LDM	
+1	BN3	Users current bank
+2	STF	3
+3	LDM	
+4	SA	
+5	0	
+6	CIL	Since the jump will be made directly, the interrupt lockout set by RECOV is removed.
+7	INSTRUCTION	

Procedure:

a. Store CD1 in T5+7.

b. Call GEN with calling sequence 8, T5.

c. Go to ALPHA.

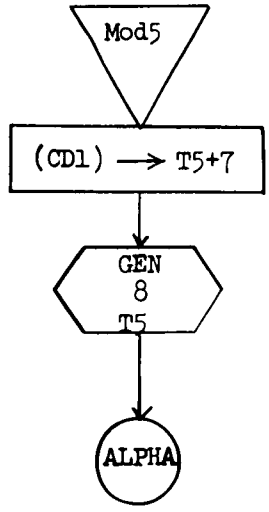

Figure 7.35 MODULE 5

7.3.7.3.6 MODULE 6 (Figure 7.36)

This module translates the bank set instructions. The translation sequence is:

T6	INSTRUCTION	
+1	CTA	
+2	SIC	SFCHEX BANK
+3	STM	
+4	BC	Bank settings for return to user's program.

Procedure:

a. Store CD1 in T6.

b. Call GEN with calling sequence 5, T6.

c. Go to ALPHA.

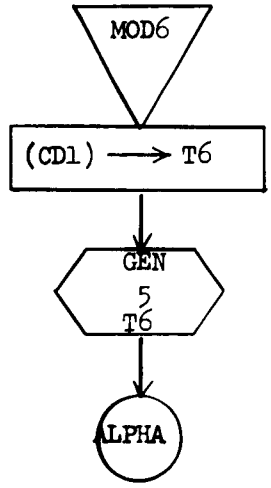

Figure 7.36 MODULE 6

7.3.7.3.7 MODULE 7 (Figure 7.37)

This module translates BLS, ATE and ATX instructions. The translation sequence is:

T7	INSTRUCTION		
+1	*+3		Just jump to T7+4 if exit occurs.
+2	JFI	1	
+3		12	
+4	SIC	SFCHEX BANK	
+5	LDC		
+6		G	Obtained by WT
+7	STM		
+8	EXIT		
+9	JFI	1	
+10	A32		

Procedure:

a. Store CD1 in T7.

b. Store the address in CD4 increased by 4 in T7+1 for the jump to T7+4 in the execution table.

c. Call WT with T7+6 in the A-Register.

d. Call GEN with calling sequence 11, T7.

e. Go to ALPHA.

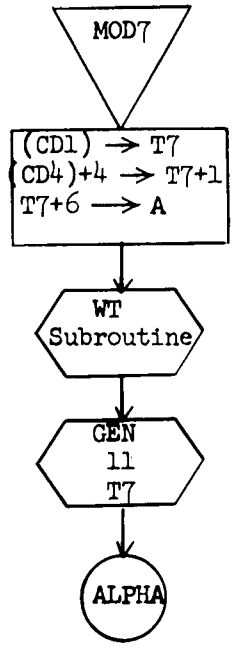

Figure 7.37 MODULE 7

7.3.7.3.8 MODULE 8 (Figure 7.38)

This module translates the STP Instruction. The translation sequence is:

```
T8    STF    5      Save A-Register
+1    LDC
+2           (P)
+3    STD    5X
+4    LDC
+5           -
```

Procedure:

a. Store the address of P contained in CD5 in T8+2.

b. Add 4000 to CD3 and store in T8+3.

c. Call GEN with calling sequence 6, T8.

d. Go to ALPHA.

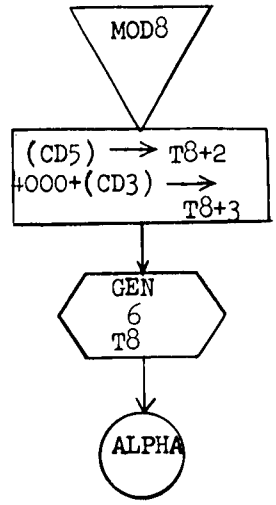

Figure 7.38 MODULE 8

7.3.7.3.9 MODULE 9 (Figure 7.39)

This module translates the PTA instruction. The translation sequence is:

```
T9    SIC    SFCHEX BANK
+1    LDC
+2           (P)
+3    STM
+4    SA
```

Procedure:

a. Store the address in P contained in CD5 in T9+2.

b. Call GEN with calling sequence 5, T9.

c. Go to ALPHA.

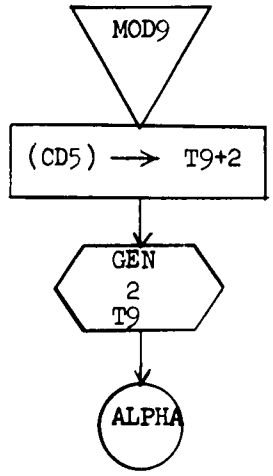

Figure 7.39 MODULE 9

7.3.8 UNST Subroutine (Figure 7.40)

This subroutine may be called with the parameters stored in the BUF Buffer or a TRAP Table. In either case the parameters are stored in the following order:

A1	bank 1
A2	address 1 (to place the jump to RECOV)
A3	bank 2
A4	address 2 of snap command to be removed

Procedure:

a. Set T3+1 to 6 and T1 to 1. This is for INSV Subroutine.

b. Store -5 in the TRAP Table via DTR. This will cause the next trap execution to skip this parameter. If UNST was called from the BUF Buffer no harm will be done.

c. Call INSV subroutine.

d. Return.

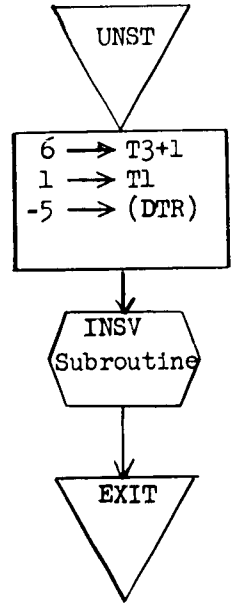

Figure 7.40 UNST

7.3.9 Mnemonic Dump Subroutine (Figure 7.41)

This subroutine can be called directly from the input section or from a trap table. The sequence of setup parameters is the same in each case.

A1	bank	- bank number
A2	address	- start of dump
A3	zero	- unused
A4	number	- number of 60 word blocks

The procedure is:

a. The address A1 is contained in DTR. The Bank number is stored into DUMAD via D.

b. Increase DTR and D1 by 1. Store address in DUMAD+1 via D1.

c. Increase DTR + D1 by 1. Store 0 into DUMAD+2 via D1.

d. Increase DTR + D1 by 1. Store number into DUMAD+3 via D1.

e. The UNNEED Subroutine is called in case the dump is in the direct bank.

f. The line is output.

g. DUMAD+1 is increased by 60 and stored in the output command.

h. DUMAD+3 is decreased by 1.

i. Test DUMAD+3
   0 - Call NEED Subroutine and exit.
   ≠ 0 - Go back and print next image.

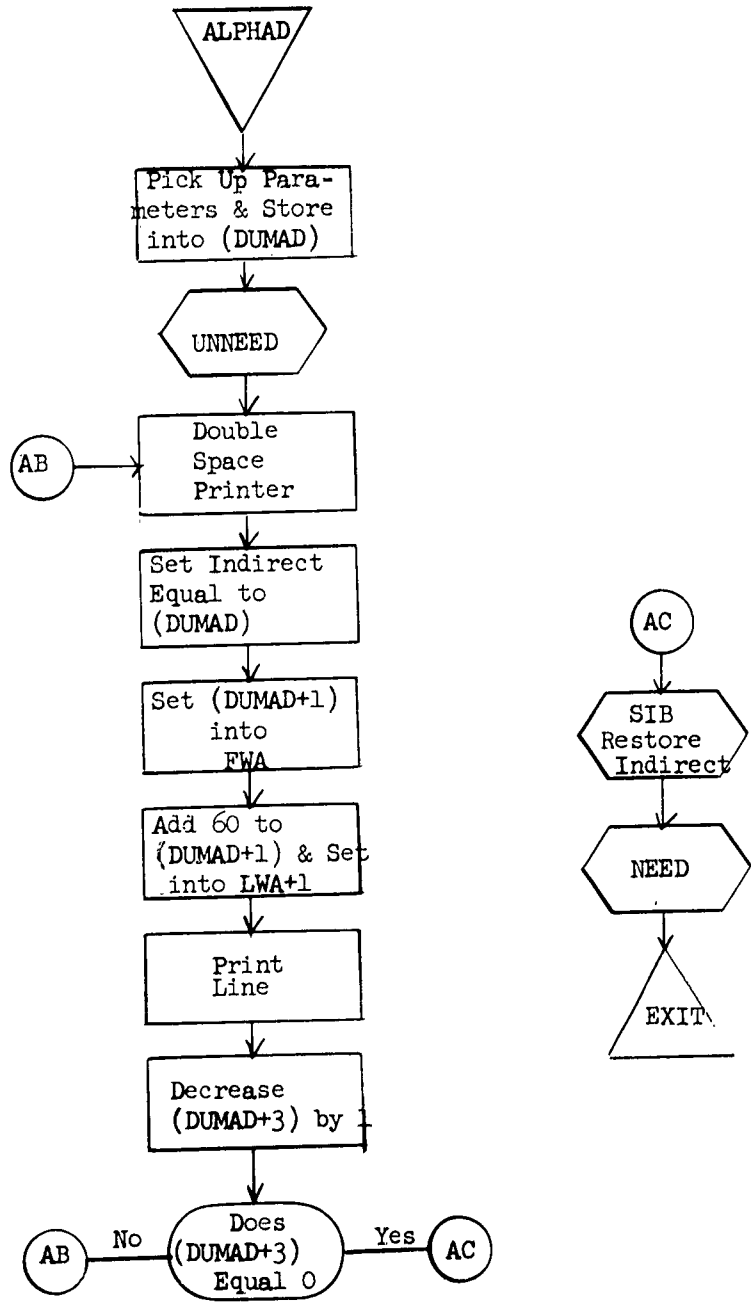

Figure 41 Mnemonic Dump Subroutine

7.4 ENTRY Subroutine - TRAP, SNAP or UNSNAP Execution (Figure 7.42)

Procedure:

a. Store bank settings in BC.

b. Call NEED Subroutine to save user's direct cells and to store SFCHEX' direct cells.

c. Set indirect bank to user's relative bank.

d. Find address of flagged jump +2 and store in DP.

e. Pick up contents of user's A-Register and store in TSA.

f. Put final user's bank controls in BC.

g. Transfer DP to EXIT, decrease DP by 2, store TSA in SA.

h. Set DE2 to 0. Store the address of TAB Table in DTAB.

i. Store -1 in DE1.

j. Compare the address in the TAB Buffer (via DTAB) with the address in DP. If equal go to step m, otherwise continue.

k. Increase DTAB by 3, increase DE1 by 1.

l. If DE1=0, return to step 10.
If DE1=1, increase DE2 by 1 and go to step 9.

m. At this time we have found the matching address. The contents of DE2 will tell SFCHEX what kind of command is to be dealt with.

DE2=0	TRAP
DE2=1	UNSNAP
DE2=2	SNAP

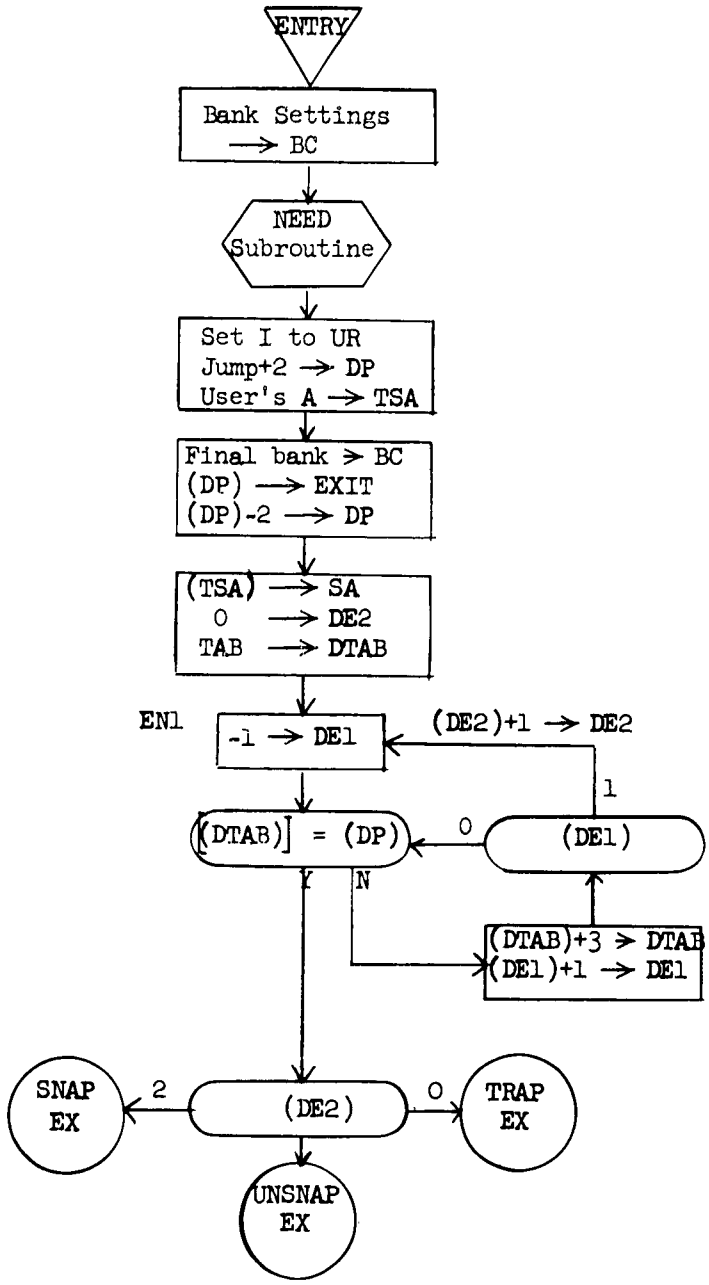

Figure 7.42 ENTRY

7.4.1 TRAP Execution Section (Figure 7.43)

This section locates the proper TRAP TABLE (in TT Buffer) and TRAP Instruction Execution Table (in TX Buffer) and then executes them.

Procedure:

a. Store initial address of TX Buffer in DE4. Store initial address of TT Buffer in DTR.

b. If DE1 equals zero, go to step d, otherwise continue.

c. Increase DE4 by 30, DTR by 50.

d. Are the contents of the address in DTR equal to -1. Yes - this is an infinite trap, go to step e. No - decrease the contents of DTR by one via DTR.

e. Store the contents of the address in DTR in EN12+1. Increase DTR by one.

f. Check the contents of the address stored in DTR for 7777. If 7777, the TRAP execution is finished, go to step 10. If not 7777, test the value for less than 0 and greater than -20. If this condition doesn't hold, go to step i.

g. If the value is -M, where M is less than 20, this means that a SNAP, UNSNAP or CORRECT command was executed previously for this TRAP Table.

h. Add M to the address in DTR. Return to step f.

i. DTR contains the address of some subroutine (DUMP, HEAD, UNST, SNST, CORRECT). Transfer to that subroutine. (Note the subroutine will modify DTR so that it contains the next address to be considered in the TRAP Table.) When subroutine returns control, return to step f.

j. The TRAP Table has been executed. Check the value in EN12+1 (the TRAP iteration counter). If 0, the trap has been flushed, go to step k. Otherwise return to the user's program via PREX which will execute the instructions at the address contained in DE4 (in TX Buffer).

k. Call REPLAC (Section 7.4.5). This subroutine replaces the contents of the cells which were taken from the user's program. Switches are set so that the TRAP command is obliterated.

l. Test NOPUT Switch. 0 - more input - go to BETA and input next commands. 1 - no more input - continue.

m. Test TERM Switch.

1 - go to step n.
0 - call UNNEED Subroutine and go to A32 in PREX for return to user's program.

n. Test DE1

If -1, test the contents of TT+50
If 0, test the contents of TT.

(Since one trap has just flushed, we need only check the N cell in the other TRAP Table for 7777 or 0.)
If 7776 or 0, stop.
Otherwise call UNNEED Subroutine and go to A32 in PREX for return to user's program.

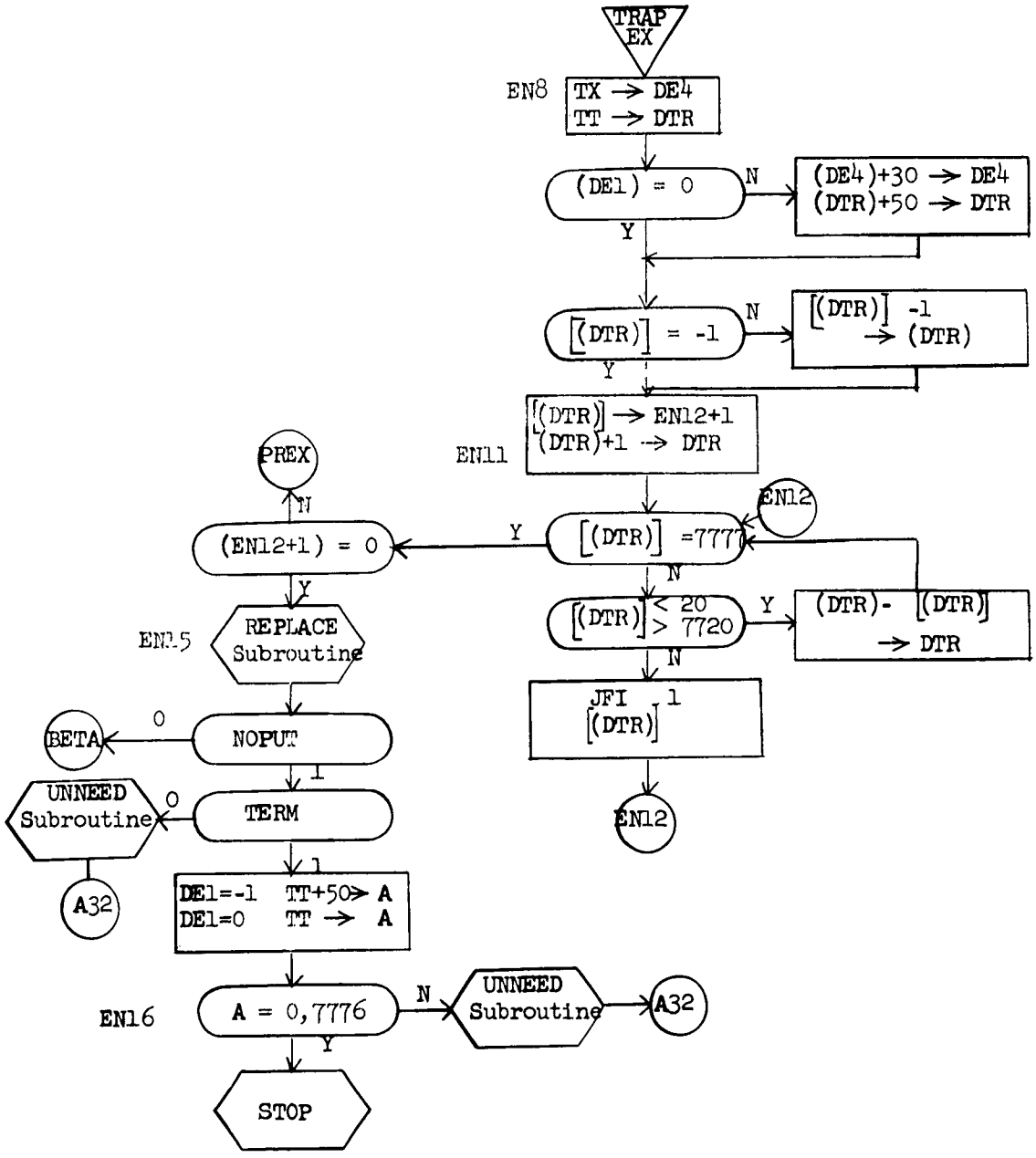

Figure 7.43 TRAP EX

7.4.2 UNSNAP EX (Figure 7.44)

This section removes a snap address from the TAB Buffer and replaces the saved instructions in the user's program at both the SNAP and UNSNAP jump locations.

The contents of DE1 will tell which of the addresses in the UNSTBL is to be used in removing the SNAP COMMAND.

Procedure:

a. Call REPLAC. DTAB will indicate the set of cells and corresponding address which are to be replaced. This takes care of the UNSNAP jump address.

b. Set the initial address (plus 12) of TAB in DTAB. Pick up the SNAP address from UNSTBL+DE1 and store it in CD1.

c. Compare the address in UNSTBL (via CD1) with the cell in TAB whose address is in DTAB. If equal, then the address is found, go to step e. If unequal, increase DTAB by 3 and continue.

d. Test DTAB. If equal to 18, an erroneous UNSNAP command was entered, since no match can be found. The following message will appear on the typewriter. "NO S AT BXXXX". Then SFCHEX will halt. Otherwise return to step d.

e. Call REPLAC. The saved instructions will be stored at the SNAP jump address.

f. Go to A32.

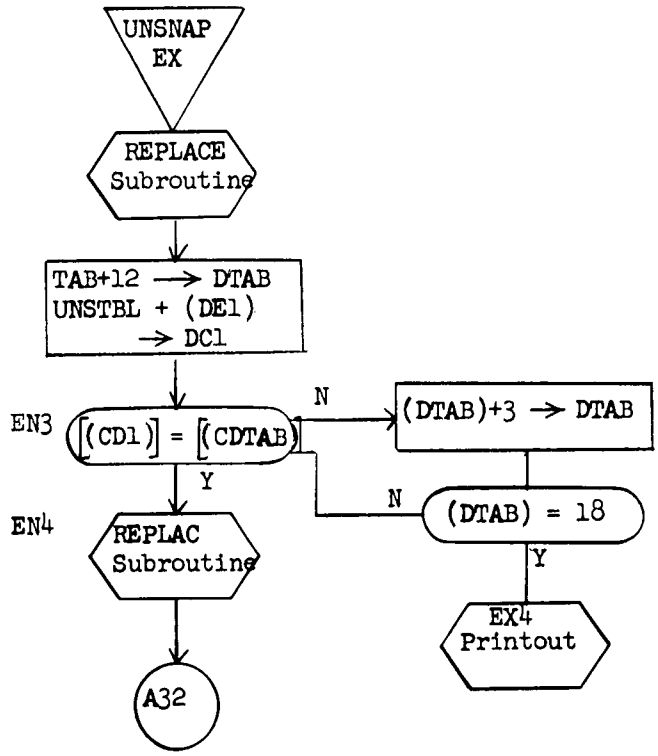

Figure 7.44 UNSNAP EX

7.4.3 SNAP EX (Figure 7.45)

This section sets up the execution of the SNAP dump.

Procedure:

a. Pick up the snap bank from address SNPTBL+(DE1). Store the result in CD1.

b. Call SNAP (Section 7.4.6) - a snap dump ensues.

c. Put the initial address of SX in DE4.

d. Test DE1 - 1 - DE4 continue the address of the right translation sequence. Go to PREX.

e. Increase DE4 by 30, and go to PREX.

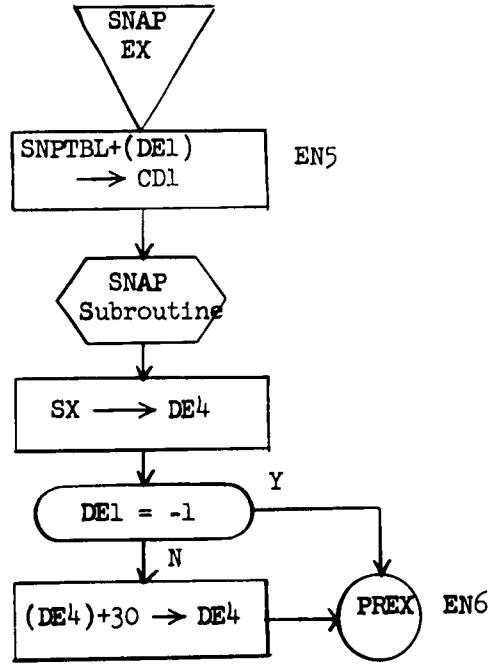

Figure 7.45 SNAP EX

7.4.4 PREX (Figure 7.46)

This subroutine has two entry points. The main entry (PREX) causes a translation sequence (initial address in DE4) to be executed. A32 is entered when no instructions are executed. This will result in the A-Register and banks being restored and a return made to RECOV.

Procedure;

a. Call UNNEED Subroutine in order to return user's direct cells.

b. Set the banks and A-Register as they were in the user's program.

c. JFI to the address in DE4.

Note: There are two possible returns from step c. A31 or A32.

d. Store the present contents of the A-Register in SA.

e. Store the contents of the SA in the RECOV Subroutine.

f. Store the contents of EXIT in RECOV.

g. Set the banks and jump to RECOV.

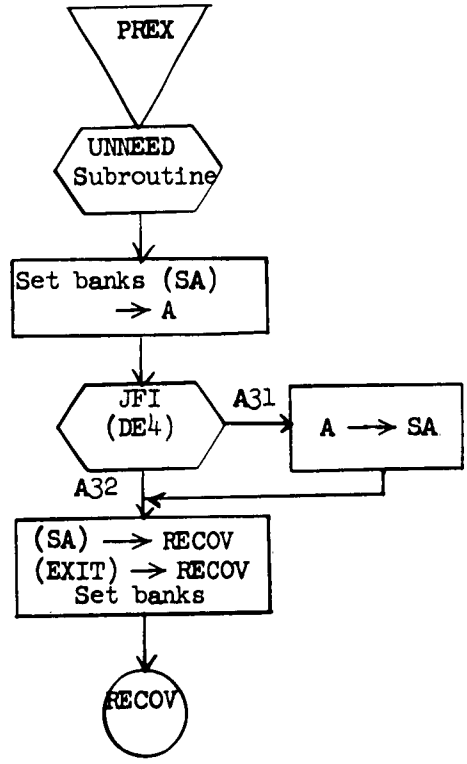

Figure 7.46 PREX

7.4.5 REPLAC (Figure 7.47)

This subroutine replaces the instructions which were saved in the TAB Buffer as the result of a SNAP, UNSNAP, or TRAP command.

DTAB will contain the address of the location in TAB of that three cell group containing an address in the user's program and the 2 saved cells.

Procedure:

a. Pick up the address stored in DTAB and store it in CD1.

b. Pick up the user's program address via CD1 and store it in CD2 and EXIT.

c. Store a 7777 in TAB via CD1. This will allow subsequent commands to use the space.

d. Increase CD1 by one. Pick up the first saved instruction in TAB and store it in CD3.

e. Increase CD1 by one. Pick up the second saved instruction in TAB and store it in CD4.

f. Set the indirect bank to the user's relative.

g. Store the contents of CD3 in the user's program via CD2. Increase CD2 by one, and store the instruction in CD4 in the user's program.

h. Reset the indirect bank and exit.

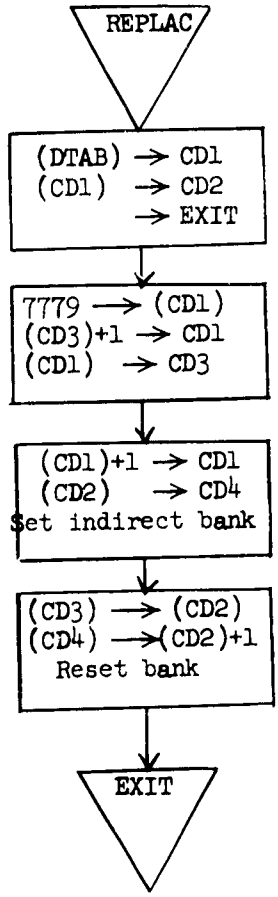

Figure 7.47 REPLAC

7.4.6 SNAP (Figure 7.48)

The SNAP Subroutine sets up a special calling sequence for the DUMP Subroutine. The bank to be snapped will be in the A-Register. If the A-Register contains a 4, all the banks will be snapped.

Recall the DUMP Calling Sequence			Parallel SNAP Sequence		
A1	bank 1		SNPBS	----	bank
A2	address 1	Lower Limit		0	
A3	bank 2		SNPBT	----	(same bank)
A4	address 2	Upper Limit		77	

Procedure:

a. Store A-Register in SNP10.

b. Does A-Register equal 4.
No - go to step d.
Yes- continue to step c.

c. Store 3 in SNP10 and set 0 in A-Register.

d. Store A-Register in both SNPBS and SNPBT and store the address SNPBS in DTR for the DUMP Subroutine.

e. CALL DUMP.

f. Subtract SNPBS from SNP10.
0 -RETURN
Non-0-go to step g.

g. Increase SNPBS by 1 and leave the result in the A-Register. Go to step d.

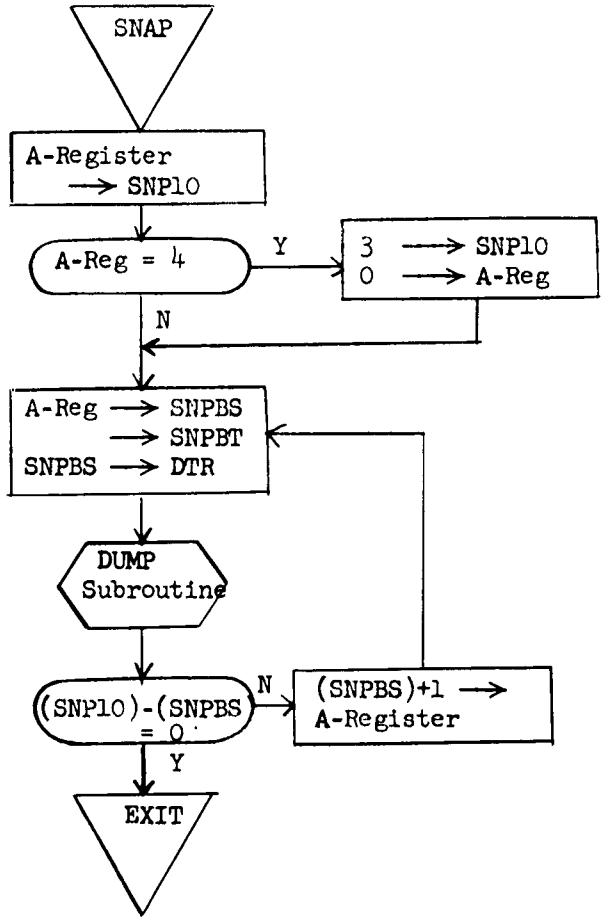

Figure 7.48 SNAP

DISTRIBUTION LIST

EXTERNAL

Space Systems Division
(Contracting Agency)
Maj. C. R. Bond (SSOCD)

6594th Aerospace Test Wing
(Contracting Agency)
Lt. Col. A. W. Dill (TWRD)
Lt. Col. M. S. McDowell (2)
TWACS (6)
V. Thomas

PIR-E1 (Lockheed)
N. N. Epstein
C. H. Finnie
H. F. Grover
H. R. Miller
W. E. Moorman
461 Program Office
698BK Program Office

PIR-E2 (Philco)
J. A. Bean
J. A. Isaacs
R. Morrison
S. M. Stanley

PIR-E3 (LFE)
D. F. Criley
K. B. Williams (5)

PIR-E8 (Mellonics)
F. Druding

PIR-E5 (Aerospace)
F. M. Adair
R. V. Bigelow
R. D. Brandsberg
L. H. Garcia
G. J. Hansen (3)
C. S. Hoff
L. J. Kreisberg
T. R. Parkin
E. E. Retzlaff
H. M. Reynolds
D. Saadeh
R. G. Stephenson
V. White

PIR-E7 (STL)
A. J. Carlson (3)

PIR-E4 (GE-Sunnyvale)
J. Farrentine
N. Kirby

PIR-E4 (GE-Santa Clara)
D. Alexander

PIR-E4 (GE-Box 8555)
J. S. Brainard
R. J. Katucki
J. D. Selby

PIR-E4 (GE-3198 Chestnut)
J. F. Butler
H. D. Gilman

PIR-E4 (GE-Bethesda)
A. Pacchioli

PIR-E4 (GE-Box 8661)
J. D. Rogers

12 February 1963

TM-1003/002/00

Internal Distribution List

G. Wilson	22101
M. Winsor	24137
J. Winter	24097
R. Wise	24051
J. Wong	Sunnyvale
R. Busch	24065

12 February 1963

TM-1003/002/00

Internal Distribution List

D. Allfree	22078	R. Keyes	20073
J. Aldana	24113	J. Kneemeyer	24065
B. Alexander	22083	R. Knight	24110
N. Alperin	24118	R. W. Knight	22095
E. Armstrong	24089	L. Kolbo	24139
C. Becerra	24082	L. Laughlin	20073
D. Biggar	24090	J. LaVine	20079
R. Bilek	24124	H. Lewis	24117
L. Brenton	22070	J. Little	20077
B. Burke	23014	F. Long	24122
R. Burke	23014	J. Lytton	24077
C. Bustya	22084	G. Madrid	24049
M. Champaign	24127	G. Mahon	22076
D. Chesler	22087	R. Marshall	24117
C. Chiodini	22078	J. Marioni	24076
B. Ciaccia	24082	W. Martin	24089
R. Clements	24132	J. McKeown	24121
B. Cline	24097	J. Milanese	24121
J. Cogley	24135	J. Munson	24048
L. Conger	22079	G. Myers	14056
P. Cooley	24083	P. Nelson	24075
D. Crum	24093	L. Ngou	25030
L. DeCuir	22096	M. Olson	24124
W. Derango	24082	L. Padgett	24085
G. Dexter	24128	E. Patin	Sunnyvale
R. Disse	24139	D. Persico	20076
G. Dobbs	24094	T. Polk	24103
W. Dobrusky	22125	D. Reilly	24085
R. Dugas	24105	M. Rockwell	22070
R. Ellis	24081	C. Seacat	Sunnyvale
R. Ericksen	24110	H. Seiden	22091
H. Feldstein	27013	R. Scott	24093
C. Francis	20075	R. Shapiro	25036
M. Franks	25030	S. Shoel	24123
L. Friedman	22083	R. Skelton	24127
S. Gardner	22053	N. Speer	20079
V. Gergen	24109	E. Stone	22116
I. Greenwald	24058	M. Sweeney	24057
J. Haake	24120	W. Taber	22053
D. Henley	24058	T. Tennant	27024
C. Hill	24057	J. Thompson	22077
J. Hillhouse	24049	C. Toche	24088
H. Holzman	22096	R. Totschek	24090
G. Hudson	22101	A. Tucker	24115
R. Johnson	24105	A. Vorhaus	24076
P. Kastama	24053	S. Weems	24126
M. Katz	24103	G. West	Sunnyvale
F. Kayser	25026	G. P. West	24094
J. Keddy	25026	M. Weinstock	22095
D. Key	24123	B. Williams	24091

UNCLASSIFIED

System Development Corporation,
Santa Monica, California
MILESTONE 11 160-A DIAGNOSTIC
PROGRAM (SFCHEX)
Scientific rept., TM-1003/002/00, by
Utility Section. 12 February 1963, 159p.,
48 figs.
(Contract AF 19(628)-1648, Space Systems
Division Program, for Space Systems
Division, AFSC)

Unclassified report

DESCRIPTORS: Satellite Networks.
Programming (Computers).

Reports that SFCHEX (160-A Diagnostic
Program) allows the user to dump any

UNCLASSIFIED

---

portion of his program at any time,
modify parameters or instructions as
he desires and to jump from point to
point in his program. SFCHEX performs
these functions by accepting directions
via the typewriter or on-line card reader.
Considers the input and output formats
and the operating instructions of the
program.

UNCLASSIFIED

UNCLASSIFIED